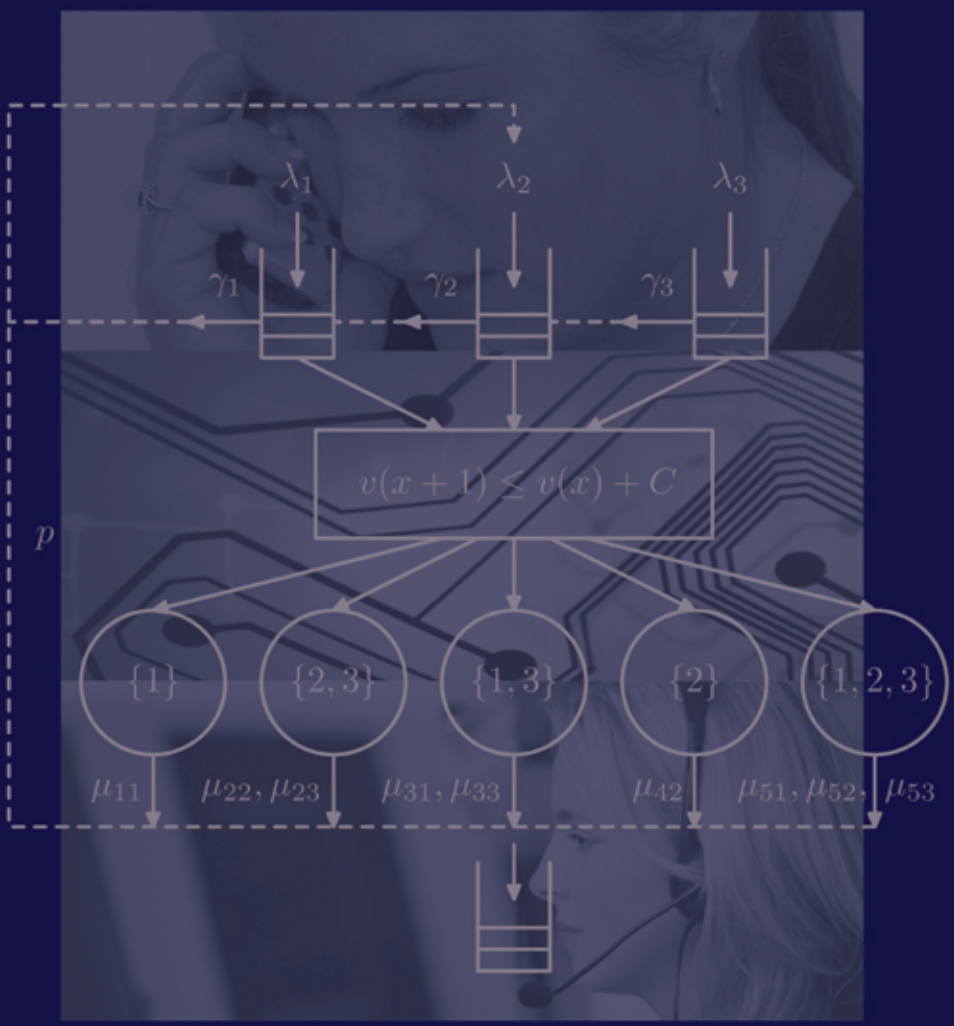
Planning and Routing Algorithms for Multi-Skill Contact Centers

Pot, S.A.

2006

document version
Publisher's PDF, also known as Version of record

Planning and Routing Algorithms for Multi-Skill Contact Centers



Auke Pot

Planning and Routing Algorithms for Multi-Skill Contact Centers

Pot, Simon Auke, 1979
Planning and Routing Algorithms for Multi-Skill Contact Centers
ISBN-10: 90-8659-023-3
ISBN-13: 978-90-8659-023-0

Thomas Stieltjes Institute
for Mathematics

Printed by PrintPartners Ipskamp, The Netherlands.

VRIJE UNIVERSITEIT

Planning and Routing Algorithms for Multi-Skill Contact Centers

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad Doctor aan
de Vrije Universiteit Amsterdam,
op gezag van de rector magnificus
prof.dr. L.M. Bouter,
in het openbaar te verdedigen
ten overstaan van de promotiecommissie
van de faculteit der Exacte Wetenschappen
in de aula van de universiteit,
De Boelelaan 1105

door

Simon Auke Pot

geboren te Hoogeveen

promotor: prof.dr. G.M. Koole

Preface

The last four years of my work have been satisfactory; due to the research I was able to deploy and to learn a lot.

In the first place, I would like to thank my advisor professor Ger Koole for allowing me to do this, and for giving me support, feedback, freedom, and nice opportunities.

I am also indebted to the reading committee (consisting of Sem Borst, Philippe Chevalier, Shane Henderson, and Rob van der Mei) for their effort. Their comments on the manuscript were very useful.

Additionally, I mention several colleagues that played a crucial role. Sandjai Bhulai and Geert-Jan Franx have been very helpful by their professional feedback on my work and by the interesting discussions about research.

Furthermore, Menno Dobber and Maarten Soomer made the period very enjoying, not only at the university but also during holidays and evenings in Amsterdam.

Next, my family played an important role: by their love and showing their interest in my work. In particular, I thank my parents Jan and Jeanet for stimulating me to start as a Phd. student, and Demian and Lianne for giving me feedback on my writing.

Finally, I would like to thank Marit for her love, care, and enthusiasm. Last years of my personal life have been very pleasant due to her, and, meanwhile, her discipline helped me to continue focusing on my research. I also thank her for proofreading the manuscript of this thesis.

Auke Pot
July 2006

Contents

Chapter 1

Introduction

During the last decennia call centers have become a well-known phenomenon. The total number of call centers has increased substantially and call centers have become bigger: the larger call centers now employ five hundred up to two thousand employees, there are ten thousands of call centers across the globe, and between one and half million and two million employees. Outsourcing to for example India is becoming very popular. One expects that this country will build up an industry that is worth seventeen billion by 2008. In India hundreds of thousands of engineers who can work in call centers, graduate each year, see NASSCOM (2006). A reason for the growth of the industry is that customer relationships and customer services have become more important. There is also an explanation for the fact that call centers have become larger; more representatives are assembled together in one building to improve service, reduce costs, and obtain a higher productivity. However, maximizing the economies of scale also requires improvement of the operational processes, especially concerning job routing and workforce management (WFM). Hence, forced by the pressure of improving these processes, new companies have arisen that support the call centers by delivering software, hardware and consulting services. The developments that concern the hardware and the corresponding software such as routing devices and databases are of central importance to the developments of the industry. The popularity triggered researchers to work on many different subjects. The work of mathematicians on planning and job routing is the subject of this thesis.

Contact centers

In this thesis we focus on a special type of contact center, namely multi-skill contact centers. The difference between contact centers and call centers is that in contact centers also other channels than telephony are used, for example fax and email. We will explain the meaning of the term 'call center' first. Mehrotra (1997) defines a call center as "any group whose principal business is talking on the telephone to customers or prospects". The employees talking on the telephone are commonly called agents, telephone service representatives, or customer service representatives (CSRs). Similarly, we define a contact center as any group whose principal business is communicating to customers or prospects. The term 'multi skill' will be explained in Section 1.1.

Service level

Many problems concerning the routing of jobs and planning of employees are restricted by service-level (SL) constraints. Different factors determine the quality of a service that is offered to a customer, e.g., whether or not the customer has been served, whether or not the customer was rejected, the voice of the employee, etc. We note that some of these factors are difficult to measure. In practice, call centers do not take all of them into account when optimizing and scheduling the operational processes of workforce management. Both the quality of the service itself and the waiting-time distribution are considered as very important quantities. An example of a service-level measure often used in call centers is the percentage of all customers that are served within twenty seconds waiting. Eighty percent is considered as an acceptable level, in the sense that the costs of employees and the service level are well balanced. In addition, companies also like to monitor abandonments, rejections and blockings, but these are less important and not always used for planning and optimization. The different service-level measures are related to each other. For example, short waiting times often imply low abandonment and blocking percentages.

Economies of scale

Much work in the literature is related to the size of a call center. The size is an important quantity, because of the relation with costs and productivity. In general costs decrease relative to the size and productivity increases when

call centers grow. These economic advantages are the so-called economies of scale. Also much literature about routing and planning pays attention to the economies of scale. The papers from the literature often try to characterize and quantify the relation between on the one hand routing policies and planning methods, and on the other hand the economies of scale. Hence, the economies of scale are important to the subject of this thesis and deserve to be discussed in more detail. This is done in Section 1.3.

1.1 Trends and growing diversity

There are many different types of contact centers. We partition the types by discussing the different ways in which calls can be initiated and we describe recent trends. In addition, the differences in routing and planning are explained. Besides, the section illustrates that routing and staffing in inbound call centers are difficult issues and, hence, are the main subjects of the remainder of this thesis.

1.1.1 Traffic types

In this section we make a classification in the way calls are initiated, the so-called traffic types. It describes the trend from outbound call centers to inbound call centers, as occurred in the eighties and nineties. For each type we address the routing and planning problems and explain the differences.

Outbound

In outbound call centers agents used to dial manually. In general, each call is served by exactly one agent, i.e., the agent that initiated the call. Thus, the assignment of work to agents is straightforward and there is always a match between the workload and the capacity of the agents. Hence, routing is not relevant to this type of outbound call centers.

The planning of agents, in particular shift scheduling and rostering, is easy. The agents generate their own work such that a productivity of one hundred percent can easily be obtained, assuming that sufficient telephone numbers of potential customers are available. Another relevant issue is to find a match between the scheduled number of agents and the number of agents required to meet economical objectives, while taking the preferences of the agents into account. In many realistic cases straightforward deterministic

techniques for workforce management will suffice, because there is hardly or no randomness involved.

Within the class of outbound call centers a further classification is possible. If calls are initiated by the system, instead of agents, we speak of predictive dialing. Otherwise we speak of the classical type of outbound calling.

For example, predictive dialing occurs in call centers that approach their potential customers via telephone. A potential customer hears a recorded message and can decide to have talk to an agent. Thus, the searching process for new customers often occurs completely automatically. In this way, agents speak to interested customers out of the pool of potential customers only. Usually, only a small percentage of the potential customers is interested. We say that there is a probability of success involved in each call and we can distinguish successful and unsuccessful calls. An important characteristic of predictive dialing is that its effectiveness depends on the hour of the day that one calls. This type of outbound traffic is called predictive dialing. Just like in inbound call centers, the epochs at which interested customers are found is a stochastic process that depends on the number of lines in use for searching. Nevertheless, these processes are not exactly the same. An essential difference with inbound call centers is that the number of lines in use for searching is controllable (recall that the calls are initiated by the system), by which it is possible to anticipate successful calls that occur in the near future. We can ask the same questions concerning inbound call centers: Idleness, over-staffing and waiting times play also an important role in this type of call centers.

In Samuelson (1999) a heuristic method is described to determine the optimal number of lines to initiate. The model exploits general service time distributions because estimates of the remaining service times are used. With that data it is possible to make better decisions about the number of new lines to initialize. They report that it performs well.

In call centers that use predictive dialing, agents usually have a single skill. For example, agents do one campaign after the other. Furthermore, they are trained in advance such that one can assume that there is no dependence between agents and service rates. In case of a single skill, numerical methods for routing and staffing are easy to derive by standard techniques.

Inbound

In inbound call centers, a call is initiated by the customer. The call is for that reason called an inbound call.

Nowadays, most call centers treat inbound traffic. Inbound call centers support (existing) customers, e.g., customers having questions about their products, people having products that need repair, and customers that aks to execute transactions concerning financial products. Thus, these call centers provide different types of services, e.g., supplying information and selling.

The time epochs at which an inbound call center receives calls are random in time. This complicates the staffing of agents. Typically, a high productivity is obtained by staffing exactly the number of agents required to handle all work. However, the service level will be low and waiting times will be long. Hence, to meet a service-level requirement it is necessary to staff additional agents, called safety staffing.

1.1.2 Trends

This section illustrates the growing diversity of call centers by treating: the mixture of jobs from different traffic types, the difference between cost and profit centers, and the growing importance of multi-skill call centers.

Call blending

Currently, the telephone is not the only channel for the communication between companies and its customers anymore. Nowadays, fax, and increasingly email, represents a substantial part of all work. In addition, other types of services have gained popularity. Especially the services by which customers are served without human interaction, for example by means of video and recorded messages offered via internet, often called self-internet services. Therefore, we no longer speak of call centers, the term 'contact center' is more appropriate. The variety of mathematical models has increased significantly during recent years, because jobs from the different channels require different qualities of service (called service levels). This thesis treats routing in the broad sense; it not only focuses on telephone services but also on the other types of communication between call centers and customers.

We speak of call blending when agents work on inbound calls as well as on other activities, such as emails, faxes, or outbound calls. Typically, the emails, faxes, and outbound calls have a lower priority than inbound calls.

For that reason agents only start these blending activities when the workload of the call center is low and agents are idle. To determine the optimal number of agents working on the blending activities it is important to anticipate calls that will arrive in the near future. Consider for example call blending of inbound and outbound traffic and inbound having higher priority than outbound traffic. Then, at busy moments it might be attractive to interrupt an outbound job to handle a high-priority job. This would increase the service level of the high-priority jobs. In practice, call centers avoid that inbound or outbound calls are interrupted too frequently. If agents have to switch between the jobs too often, the productivity and service levels of both types of calls might decrease by the switching times, see for example Chapter 7 of Koole (2005).

Cost and profit centers

Most call centers are part of a large organization and their primary role is supporting customers in using their products or services. An example is the helpdesk of an internet provider. Because these centers only incur costs and do not generate income directly, they are so-called cost centers.

If rewards can be attributed to calls, and if these rewards compensate the costs, then we speak of a profit center. An example is a call center that belongs to a sales department, because selling products or services are commercial activities. A second example is a call center that handles calls for other call centers that outsource work.

There are only a few papers on outsourcing in the literature. In Koole and Pot (2006a) a method is discussed to optimize profit by controlling the number of lines and agents.

Multiple skills

In single-skill call centers there is no distinction between the handling of different call types. Agents are supposed to handle all types of calls.

We speak of multi-skill call centers in case of different job types and agents having multiple skills. An example of a multi-skill call center is an insurance company that receives damage reports about cars, houses, boats, and so forth. It is imaginable that some agents are trained to handle jobs about exactly one of these subjects. These agents are called specialists. Agents that are specialized in multiple skills are called cross-trained agents.

Finally, agents that can handle all call types are called generalists or fully cross-trained agents.

Implementing single or multiple skills in a call center is a tactical decision. A single skill call center can be inefficient if agents need much knowledge in order to handle calls. Two reasons are given. In the first place, the training of new agents can be time consuming because they need to know a lot before they can start working. In the second place, it has been shown that service times increase and productivity decreases when agents handle calls with a high diversity of subjects. Hence, in case of long handling times it might be attractive to introduce multiple skills. Then the required knowledge and capabilities of agents are split into several classes, called skills. The benefit is that agents do not need all skills to become operational. This reduces education and training costs. Whitt (1999) investigates the trade-off between the economies of scale associated with larger systems and the benefits of assigning different types of customers, with regard to the service-time distributions, to different agent groups.

Introducing multiple skills has some drawbacks. The costs of the technological infrastructure are high and it brings increasing complexity into the routing of jobs and planning of workforce.

1.2 Decision levels

We mention three important issues concerning job routing and workforce management: design, planning, and control. The differences between these three issues are explained and several examples are given. The examples are classified into operational, tactical, and strategic decisions. However, not all combinations of issues and decision types are discussed. For example, design involves mainly strategic decisions, instead of tactical and operational decisions.

Design

The design of call centers is concerned with structural long-term changes. It is by definition a long-term decision and mainly determined by strategic decisions. Examples of decisions about design are the layout of the building and the number of different skills that are distinguished among the agents. Long-term decisions have impact on short-term types of decisions. These are discussed next.

Planning

Planning is concerned with the scheduling of available resources in order to meet economical objectives. We distinguish four steps within the basic planning process of workforce management, namely: workload prediction, determining staffing levels, shift scheduling, and rostering. These are typically operational decisions. Workload prediction is concerned with the prediction of future workload. Staffing expresses the expected workload as numbers of required agents, which are the so-called staffing levels. Shift scheduling is the generation of shifts such that the staffing levels are met. We define a shift as a part of the day in which an agent can work. One shift can consist of multiple time intervals. Finally, rostering refers to the pairing of shifts into rosters and the assignment of the rosters to the employees. We define a roster as a set of shifts. When assigning rosters to employees their preferences and labor rules need to be taken into account.

In multi-skill call centers it is common that different agent groups are distinguished. The partitioning of agents into groups occurs usually in such a way that agents from the same group have (almost) the same set of skills.

In multi-skill call centers the determination of staffing levels is more complicated and staffing levels require a more detailed description, as opposed to single-skill call centers. Because groups have different characteristics, the service level depends on the division of the agents among the groups. Hence, instead of a total number it is beneficial to express the staffing levels per group, such that enough capacity is available for each call type.

Groups complicate the translation of predicted workload to staffing levels. If a skill occurs in the skill set of several groups, the activities of different groups become dependent.

Control

This section discusses the daily control of the service processes in call centers. We define control as adjustments that are executed within a short time period and triggered by external factors. Control is related to WFM because it has the potential to improve service levels and reduce personnel costs. We remark that control involves operational, tactical, and strategic decisions.

For example, on the operational level staffing deals with the re-scheduling of agents when the service level is low or high. A typical tactical decision involves acquiring new agents and determining their type of contracts. The training of the current agents for new skills lies in between and belongs

therefore to the tactical or operational decisions. Decisions concerning shift scheduling and rostering are considered as operational decisions.

A subject that is closely related to staffing, is routing. An example of a control issue of routing is the optimization of policies during operations. Routing policies are described in Section 1.4.

1.3 Economies of scale

When call centers grow the productivity of workforce can increase without loss of service level. This is explained next, in an intuitive fashion. The arrival process of jobs is unpredictable. There is a continuous deviation between the expected and the actual number of arrivals during a short time interval. Let us assume that the length of this interval is of the same order of magnitude as the acceptable waiting time and consider such an interval. There is a positive probability that during this interval more work arrives than is expected on average, which can yield a low service level because customers have to wait. In small contact centers, the error between predictions and realizations can be considerable, relative to the expected number of arrivals. The reason is that random arrival processes have a relatively high variability in the number of jobs that arrive during a short interval.

The variability in the number of arrivals is often expressed as the coefficient of variation, see for example Tijms (1986b). The number of arrivals X during the interval $(0, t)$ of a Poisson process with rate λ has a Poisson distribution with parameter λt. The coefficient of variation is given by $c_X = \frac{1}{\sqrt{\lambda t}}$, i.e., the standard deviation divided by the expectation. Note that c_X is large when λ is small, as is the case in small call centers. In contrast, in large call centers the relative deviation between actual realizations and predictions becomes negligible, when considering intervals of the same length as in small call centers. Mathematically, c_X goes to zero as λ increases. Therefore, we conclude that the arrival process of the workload becomes better predictable because the randomness decreases relative to the expected number of arrivals.

Workload predictions are related to staffing levels. We consider a short time interval again. To meet service-level conditions, it is important that sufficiently many customers are served immediately, without waiting longer than a certain time. This requires that the service capacity of the agents is higher than the workload. The reason is that the exact arrival epochs of

jobs are unknown (since numbers of arrivals fluctuate over time), such that customers have to wait when all servers are busy. Scheduling additional agents in order to meet service-level constraints is called safety staffing. As a result, agents are idle during periods with fewer arrivals.

When call centers grow larger and larger, the randomness of X decreases, relative to the expected number of arrivals, and c_X almost goes to zero. Thus, the number of arrivals becomes better predictable. Eventually, to meet the service-level conditions, the staffing levels can be chosen in such a way that the service capacity almost equals the expected workload. In the limit, the service capacity can approach the workload, relative to the expected workload, yielding a productivity of almost one hundred percent. This simplifies the determination of staffing levels, because errors and the corresponding costs are relatively smaller.

In multi-skill call centers, there is an additional factor that complicates staffing. Consider a small multi-skill call center with only specialists. Since there are no agents with multiple skills, jobs of different types are served by different groups of agents and there is no dependence between the number of busy agents in the different groups. The groups behave like separate call centers. Next, we schedule generalists instead of specialists. The benefit is that more agents are available to handle each job type. This minimizes the probability of queueing and maximizes the service level. (Assuming optimal routing of jobs and that all agents having the same skills scheduled, behave similarly, i.e., with the same service time distributions.) If the service level exceeds the level that is minimally required, some employees become redundant. By cancelling their shifts or rescheduling them on other tasks the service level remains acceptable and the productivity increases.

When the size of a multi-skill call center increases and a call center becomes large, it can be beneficial that agents specialize in specific tasks and do not combine different types of activities. The reason is that limiting the number of different tasks of an agent decreases the handling time of a job. In general, this is beneficial because shorter service times increase the productivity. Hence, in very large call centers it is almost optimal to use only specialists. Moreover, routing policies become less crucial because the gain of adding flexibility has become negligibly small and therefore only specialists will be preferred.

However, in medium-sized call centers the use of only specialists or only generalists is inefficient. Consider a call center with three arrival streams and three skill types of calls: A, B, and C. There is one fully cross-trained

agent and one specialist to serve type A calls. As we explained, by using specialists instead of generalists, more customers have to wait in this example and the service level decreases. Hence, additional agents are required to meet the service-level constraints. This decreases the overall productivity. By using also generalists, high productivity can be obtained without much loss of performance and without additional workforce, due to their flexibility. Hence, to maximize the success in medium-size contact centers it is important to find a balance between: the number of specialized agents and the number of agents with multiple tasks.

Moreover, in medium-size contact centers with agents having different skills, the routing of jobs is very important. The reason is that routing policies have a relatively large influence on the service level, as will be discussed in Chapter 8. Job routing in multi-skill contact centers is called skill-based routing (SBR). Moreover, salaries also play an important role in finding the right balance between the number of specialists and generalists. Adding skills to agents by training will increase the salaries in most cases.

For results from the literature on the economies of scale, we refer to Borst, Mandelbaum, and Reiman (2004), Whitt (1992a), and also Section 3.5. Other difficulties of predicting workload are given in Section 2.2.

1.4 Routing and staffing problems

This section treats the most relevant problems concerning routing and staffing. On these problems much literature is available, which is discussed in Chapter 3.

Routing

Routing policies specify the assignment of jobs to agents. With jobs we refer to the service of calls, emails, and faxes. The routing policies are important because they have the potential to increase the efficiency of resource usage. They usually consist of two parts: agent selection and job selection. Agent selection concerns the way arriving jobs are assigned to the agents. This happens usually immediately after an arrival. In contrast, job selection pertains to the selection of a job being assigned to an agent, either directed by the system or chosen by the agent. This often occurs immediately after an agent completes a job (if there is a job present in the queues).

A special case of job routing is call routing. Call routing takes place during the arrival of a call and the assignment of the call to an agent, and usually involves computations. This is physically directed by dedicated hardware, called automatic call distributors (ACDs). Next to the hardware implementation, job types are sometimes differentiated between channels, such as fax and email, because short waiting times are required.

The requirements on the waiting times make routing (as is most frequently treated in the literature) important because of the high potential of waiting time reduction. Reconsider the example with skills A, B, and C. When a call of type A and a call of type B arrive it might be attractive to assign the type-A call to the specialist such that the other call is served by the generalist. Observe that if the type-A call had been assigned to the generalist, it would not be possible to serve the type-B call immediately. This would decrease the service level. Besides, routing potentially has other benefits. In general, routing enables contact centers to reduce also the resolution times, which is the sum of the waiting time and the service time. This is because specialists often work faster than cross-trained agents. Then maximizing the number of jobs handled by specialists decreases the average service time of a customer.

We remark that one should be careful when changing staffing levels because counter-intuitive effects are possible, see Chapter 8 for a number of counter-examples. For example, when agents handle additional types of jobs the service level increases, assuming that the service time distributions are not affected by the additional job types. Nevertheless, for certain types of routing policies it is not difficult to construct examples such that the overall service level decreases by scheduling agents on additional job types. In our opinion, routing issues are very complicated. Also the determination of staffing levels is difficult because it depends on many factors, including the routing policy.

Staffing

The optimization of routing in conjunction with staffing is a difficult problem in multi-skill call centers. The difficulty is easy to show by means of an example. Reconsider the example with skills A, B, and C. There are two agent groups: group 1 with skills A and B, and group 2 with skills B and C. The question is: What to do when a job of type B arrives? This depends on the number of available agents in each group and many other factors. This

type of problem is computationally intractable in call centers with many agents and agent groups. The problem becomes even more complicated in case of more skill types, different service times, and different priorities of calls. These factors make routing and staffing even harder.

We elaborate on several other reasons that make staffing a difficult problem. Firstly, staffing is a complicated task because there are often multiple objectives involved, for example multiple service-level conditions. Secondly, staffing is complicated by the fact that the arrival process is stochastic, i.e., the exact arrival epoch of a job is unpredictable. Thirdly, the total daily workload is hard to predict, since there are many external factors involved. For example, an insurance company for cars observes a correlation between the weather and the number of accidents. Good service levels require accurate forecasts of the workload and thus are very important. Fourthly, the productivity of an agent is not constant during the day, and is hard to predict. It is influenced by different factors. An example is shrinkage, denoting idleness due to tasks other than serving customers, e.g., meetings and breaks. Holidays can be considered as a shrinkage factor, but differ from the other mentioned factors in the sense that holidays are well predictable (and known in advance).

1.5 Overview

This section gives an overview of this thesis. Each chapter is described separately, and information about the corresponding publications is supplied.

Chapter 2 describes the aspects of multi-skill call centers that are important to staffing and routing. These are used to set up a model in such a way that it meets reality as good as possible, while keeping the mathematical complexity moderate. Chapter 3 gives an overview of routing and staffing in multi-skill contact centers. It describes a diversity of models and mathematical tools for modeling contact centers. Chapters 1, 2, and 3 are based on Koole and Pot (2006b). In Chapter 4 we discuss standard techniques from the literature that are relevant to analyzing and optimizing call centers. Chapter 5 deals with a multi-skill call center consisting of specialists and fully cross-trained agents. All traffic is inbound and there is a queue for each skill type. Our objective is to obtain good call routing policies. In this chapter we use the so-called policy iteration (PI) method. It is applied in the context of approximate dynamic programming (ADP). Usually the PI

method is used in conjunction with the exact value function, which is well-known from dynamic programming. However, standard methods to obtain the value function suffer from the curse of dimensionality, i.e., the number of states grows exponentially with the size of the call center. Therefore, we replace the real value function by an approximation, using techniques from ADP. We apply this method to the call-routing problem, yielding very good results. The content of the chapter is based on the publication Koole and Pot (2005). In Chapter 6 we consider a multi-skill call center with one group of generalists, several groups of specialists, and general service time distributions. Jobs arrive according to a time-inhomogeneous Poisson process. Our objective is to meet targets on the service levels without using specific information about the system and without having detailed information about the arrival process. This chapter introduces a simple device to control the service levels. The method easily extends to call centers with multiple agent groups and with other service-level objectives. For the paper version of Chapter 6 we refer to Jouini, Pot, Dallery, and Koole (2006). In Chapter 7 we consider multi-class blocking systems in which jobs require a single processing step. There are groups of servers that can each serve different subsets of all job classes. The assignment of jobs occurs according to some fixed overflow policy. This model is useful for optimization problems in call centers, as well as in tele-communication and computer networks. We are interested in the blocking probabilities of each class. An approximation method is presented that takes the burstiness of the overflow processes of the agent groups into account. This is achieved by assuming hyperexponential distributions of the inter-overflow times. The approximations are validated by using simulation and we make a comparison to existing approximation methods. The overall blocking probability turns out to be approximated with high accuracy by several methods from the literature, but is even more accurate by the new method. Furthermore, the individual blocking probabilities per class are significantly more accurate for the method that is introduced in the chapter. A reference to the journal version of this paper is denoted by Franx, Koole, and Pot (2006). Chapter 8 discusses the optimization of staffing levels by minimizing employee costs, taking into account the skills of agents. The method requires an estimation of the workload and can handle service-level constraints. Chapter 9 describes simple methods for shift scheduling in call centers. We describe existing methods for single-skill call centers and introduce new methods for shift scheduling in multi-skill call centers. The methods for multi-skill call centers consist of two steps. First we determine

staffing levels and next, in the second step, the staffing levels are used as input for the scheduling problem. The scheduling problem relies on a linear programming model. The methods are easy to implement and have short computation times. Solving the linear programming models requires only a fraction of a second and yields nearly optimal results. This shows that they are simple and highly accurate methods and therefore useful for different purposes, e.g., to analyze many different scenarios and to evaluate strategic decisions. Moreover, they can be part of an iterative procedure that combines shifts into rosters. Next, a number of possible extensions to the models are discussed. For example, we explain how to include communication channels other than telephony. Chapters 8 and 9 are based on Bhulai, Koole, and Pot (2006), which is related to Koole, Pot, and Talim (2003). Finally, in Chapter 10 we look back to the content of this thesis and give our final conclusions. The directions for future work will show that there are great challenges for mathematicians.

Chapter 2

Model Description

In this chapter we present a general model of a multi-skill contact center. We aim to establish a model that is not too complicated to analyze and one that approaches reality sufficiently close to be useful in practice.

2.1 Introduction

The model needs to be appropriate for developments concerning workforce management. We restrict ourselves to a maximum period of one day, which is sufficient for modeling shifts. Besides shifts, other important features of the model are the support of multiple skills, agents working in groups and having different service times, and the support of specific routing policies.

In each section we discuss a subject that is relevant to the model. Section 2.2 treats the arrival process. In Section 2.3 characteristics of agents that are assumed most relevant, are specified. Section 2.4 relates the previous two sections to each other by describing the assignment of calls to agents, in conjunction with all associated events. In practice, agents are sometimes grouped in teams, discussed in Section 2.5. In Section 2.6 we emphasize that jobs from different channels sometimes have different importance, often expressed in service-level constraints. The presence of jobs from different channels has the potential to increase productivity while meeting service-level constraints. Finally, Section 2.7 focuses on workforce management by establishing a model for planning and scheduling. In each section, we give a description of reality first and then present the corresponding part of the model.

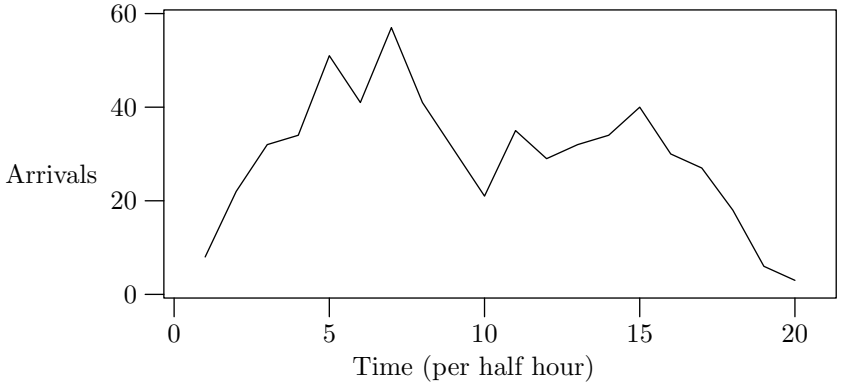


Figure 2.1: Daily pattern of work arrival

2.2 Arrival process

Several studies show that the arrival intensity of work in inbound contact centers depends on the hour of day. A typical pattern of the number of arrivals against time is plotted in Figure 2.1. The busiest hours are in the morning from 10:00 till 12:00 and in the afternoon from 13:30 till 15:30. From 19:00 in the evening till 7:00 in the morning hardly any work arrives. Of course, most of the work that arrives at night occurs via email and fax. Calls are rare at night because many call centers are closed during those hours. However, in general, the pattern can depend on different factors, e.g., the type of service, the country, and the business model.

Call centers predict the workload in order to plan agents and meet service-level objectives. Case studies show that it is difficult to make accurate predictions of workload. Although the pattern with the two peaks from Figure 2.1 is roughly every day the same in a call center, there are still big differences between the processes during different days. Not only the total number of calls fluctuates strongly, see Jongbloed and Koole (2001), but the heights and the time epochs of the peaks also differ per day. As an illustration we note that contact centers are often clearly under- or over-staffed, yielding either high or low service levels, respectively.

Predictions can be inaccurate in call centers because of randomness and unpredictable factors such as weather. Inaccurate predictions usually have much impact on service levels. For example, due to under-estimations of

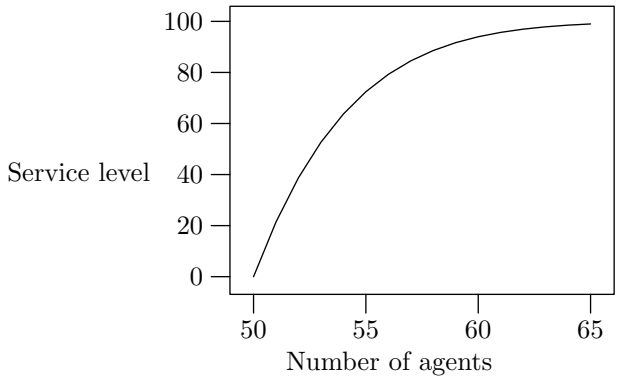


Figure 2.2: Concavity of a service-level measure

workload service levels can be expected to be low during a day. To meet service-level targets call centers have to compensate by providing service levels above the targets at other days. These compensations are undesired because of inefficiency and extra costs. We give two reasons to predict the future workload as accurately as possible, i.e., to minimize deviations between predictions and actual realizations of the workload. In the first place, long waiting times during days of under-estimations against short waiting times in case of over-estimations implies a high variance in waiting times over the different days. In the second place, costs increase in the long run because additional agents are required to meet service-level conditions. This is illustrated by means of an example. We consider two periods with a workload that requires 56 agents, in order to obtain a service level of eighty percent. The optimal number of agents (which is 56) is sometimes difficult to determine, because they are predicted. Hence, we compare two scenarios by using the Erlang-C formula:

1. due to bad predictions we schedule in the first and second period 54 and 58 agents, respectively, yielding an average service level of 77%, and

2. in case of accurate predictions, 56 agents are scheduled in both periods, which yields a service level of 80%.

Notice that the average number of agents is in both scenarios the same. However, the average service level is in scenario 1 on average below eighty

percent, as opposed to system 2 with a service level of exactly eighty percent. Obtaining a service level of eighty percent in scenario 1 requires an additional agent during one of both periods, which increases staffing costs. The underlying reason is that the service level is a concave function of the staffing level, as depicted in Figure 2.2, see also Jagers and Van Doorn (1986).

With regard to modeling, certain definitions have undesirable mathematical properties. For example, an efficient staffing algorithm by local search exists if the service level is assumed to be concave with respect to the total number of agents, see Koole and Van der Sluis (2003). However, with abandonments the concavity no longer holds. These kind of properties are undesirable because it complicates the determination of staffing levels. The service level also needs to be easily calculated for planning purposes. Performance measures that are often used by the industry are calculated by using simple queueing models. Especially, the standard Erlang-C model is frequently used to calculate measures of the waiting-time distribution. In Koole (2003) the importance of the service level definition is discussed. The paper explains that an expectation of the waiting-time W_q and the acceptable waiting time (abbreviated by AWT)

$$E(W_q - \text{AWT})^+,$$

has several benefits, for example $E(W_q - \text{AWT})^+$ is convex in W_q.

For call centers that distinguish multiple job types prediction is even more difficult, compared to single-skill call centers. The reason is that multi-skill call centers not only require estimates on all job types, but also per job type. As mentioned in Section 1.3, smaller numbers have a relatively larger variance. This explains that predictions per job type are less accurate. Moreover, studies show that the daily pattern of different job types are sometimes different, which complicates the predictions even further.

The total number of job types is M, and we define $\{1, 2, \ldots, M\} =: \mathcal{M}$. In our model, the arrival rate is denoted by

$$\lambda_m(t), \quad \text{the arrival rate of jobs of type } m \text{ at time } t,$$

in which we assume that jobs of type $m \in \mathcal{M}$ arrive according to a Poisson process with parameter $\lambda_m(t)$ at time t. This choice is often made in the literature because a Poisson process is memoryless, which simplifies the analysis. Moreover, many arrival processes are Poisson processes in practice. This is related to the Palm-Khintchine theorem, which states that the superposition of a large number of point processes looks like a Poisson process,

see form example Whitt (2002). Although it may not be correct, the arrival processes of the different types are for simplicity assumed to be mutually independent.

A priority or weight is associated with the jobs from each class

$$w_m, \quad \text{the weight of jobs of type } m.$$

We make no further assumptions on the usage of these priorities in models. Priorities are allowed to have different meanings. They can be implemented, for example, as the holding costs of queues, or as the reward of a service completion.

2.3 Agents and groups

The workload that is offered to the call center is handled by agents. We assume a one-to-one relation between job types and skills, and an agent can have one or multiple skills.

In our model we consider agent groups. We define

$$G, \quad \text{the total number of agent groups,}$$

and

$$S_g, \quad \text{the skill set of the agents in group } g,$$

with $S_g \subset \mathcal{M}$ and $g \in \{1, \ldots, G\} =: \mathcal{G}$. The number of groups and the associated skills are fixed during the day. Agents work according to a non-preemptive service discipline, i.e., the services of jobs are not interrupted and the service proceeds until the job is completed. We define a collection of working hours from $\mathcal{T} := \{1, 2, \ldots, T\}$, with T denoting the total number of consecutive periods, and

$$s_g^t, \quad \text{the number of agents in group } g \text{ in period } t, \text{ with } t \in \mathcal{T}.$$

The service of a customer usually consists of two parts: talk time and after-call work, also called wrap-up time. Talk time requires no further explanation. After-call work includes all activities related to the service of a call, like filling in a form, writing an email, and calling outbound.

The service time of an agent for a certain job is influenced by several factors. For the estimation of the service time, call centers can theoretically take as much information into account as is available, for example data

about the historical service times of the customer, the subject of the call, characteristics of the agent, etc. In our model, service times are skill- and group-dependent, and exponentially distributed with

$$\mu_{mg}, \quad \text{the service rate associated with skill } m \text{ and group } g.$$

We remark that for simplicity the working experience of the agents is not taken into account, but in reality agents often have a preferred skill.

An ambitious example of a call center model is one that takes the properties and preferences of every agent into account. Thus, every agent is considered to be unique. The call center may aim to optimize their economical objectives and meanwhile satisfy the preferences of the agents. This type of call center is a special case of our model; preferences of agents can be included by creating a different group for every agent.

2.4 Dynamics

Services of customers consist of several parts, for example dialing, waiting, and talking. We consider the service process of a call in a multi-skill call center as a dynamic process. It is in our opinion dynamic because some customers have to wait before they are served, while others are served immediately. The dynamics of the process are explained by means of a flow chart, see Figure 2.3 (we refer to Stolletz (2003) for a more extended version). The chart depicts the possible states of a call and the possible transitions between these states. For simplicity, skills and job types are disregarded in this figure. A description of the process follows next. Customers that find all telephone lines busy are blocked. Calling customers that are not blocked enter the system and reveal the topic that they need to talk about. This is for example implemented by means of an automatic-voice-response system, a press-button-menu, or a department having agents dedicated to this purpose. When the topic of a call is determined, a new job is initiated. The topic determines the type of the job. In our model we assume a one-to-one relation between the topic of the job and the skill of an agent that is required to handle the job. Thus, from the moment that the job is initiated, it is known which agents are capable to serve the customer. When one of these agents is available, it depends on the routing policy if the customer is immediately served by that agent. If the customer has to wait according to the policy, he or she ends up in a queue. In our model we also assume that

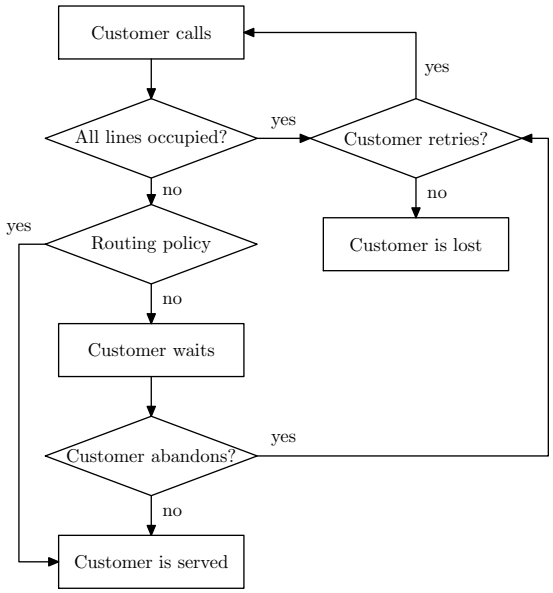


Figure 2.3: Flow chart of the service process of a customer

there exists a waiting room for every job type and

$$L_m, \quad \text{the number of waiting spots of queue } m.$$

We assume that the queue lengths are sufficiently high, such that blocking, and corresponding costs, can be ignored. Customers of each type are served according to a first-come-first-serve (FCFS) service discipline, i.e., agents choose the job that arrived earliest. Waiting customers have a patience for waiting, which we model by a stochastic variable. If the waiting time exceeds the patience, the customer abandons the system by hanging up the phone. Customers of type m have a exponentially distributed patience with parameter γ_m

$$\gamma_m, \quad \text{the parameter of the patience distribution of customers of type } m.$$

Abandoned customers may redial later when they expect shorter waiting times. We remark that due to redials the time epochs at which a customer calls are correlated, which is in contradiction with the assumption that calls arrive according to a Poisson process. Every customer who does not abandon

eventually gets to talk to an agent and is served. The agent is determined by the routing policy. It can also occur that for some reason the customer cannot be helped and is directed to another agent. This can be an agent from the back-office, depending on the difficulty of the job.

The dynamics of the call center are modeled as follows. Jobs are served by exactly one agent. Thus, redirections of jobs, e.g., to the back-office, are not explicitly modeled. However, they can be modeled by adjusting the arrival rates. Moreover, routing policies consist of two parts in our model: call-to-agent and agent-to-call. Call-to-agent policies prescribe actions at the arrival epochs of jobs, and agent-to-call policies prescribe actions at service completions, see also Chapter 1.

In our model, an email or a fax is only assigned to an agent if the agent is idle and no calls are available. The presence of emails and faxes improves the utilization of labor resources, because idle times are reduced. Idle times even disappear if sufficient emails and faxes are available. In our model we assume that faxes and emails have a lower priority than calls. Therefore, emails and faxes are served according to a preemptive service discipline, i.e., agents handling an email or fax are interrupted for calls. Due to the preemptive service discipline, the presence of emails and faxes has no influence on the service process of calls. This enables us to analyze calls without taking emails and faxes into consideration. However, we remark that preemption only approximates reality because sometimes agents complete the email or fax first, or start the service again after the interruption by a call.

Emails and faxes complicate the determination of staffing levels because the workload of faxes and emails needs to be taken into account. As well in case of preemption it is relatively simple, because the staffing levels associated with calls can be calculated independently of faxes and emails, and the expected idle times can be allocated to faxes and emails.

A graphical representation of the topology of a multi-skill contact center is depicted in Figure 2.4, a similar (but less detailed) representation is given in Cezik and L'Ecuyer (2005). The arrows at the top represent the arrival processes of the different job types and each circle at the bottom denotes an agent group. The lines in the middle represent the waiting buffers and the policy that assigns the jobs to the agents. The arrows leaving the agent groups denote that a job is completed after the service by an agent and, hence, disappears from the system.

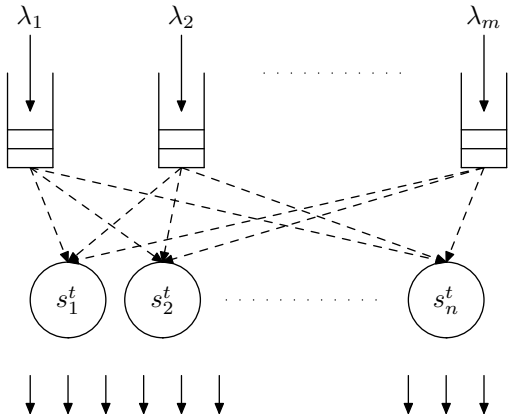


Figure 2.4: Model

2.5 Teams

In practice the way in which a call center is organized and agents are directed can influence decisions about the routing of jobs. This section describes a particular organization form and the associated routing policies, as seen in practice. We discuss their impact on the service level, especially on the productivity of agents and the waiting times of customers.

A method that has the potential to increase productivity is introducing teams. In teams agents feel more responsible for the offered work. It can avoid that agents slow down, since the whole team has to compensate for such behavior. With respect to responsibility and competition between the agents, we can expect that teams are most effective if: (1) each team consists of a moderate number of agents, (2) each skill occurs in only one team, and (3) teams are composed of agents with almost the same set of skills. It can be beneficial that each team has a leader, which can have advantages. Productivity can also increase because sometimes a competitive atmosphere arises among the different teams. However, a big disadvantage of using teams is that economies of scale are (partly) lost. Because the dependencies between the teams is minimal, every team behaves as a separate call center. Hence, routing and planning become much easier. We remark that teams can easily be translated to our model; the team setting is incorporated by taking a separate agent group for every team.

2.6 Channels, service levels and flexibility

In practice there exist several definitions of service levels. For example, the percentage of all callers that are served within twenty seconds, the average waiting time, and the percentage callers that abandons. Service-level measures can differ per channel. Usually, the handling of emails and faxes is less urgent than calls. Waiting times of several hours or even one day are often allowed, while calls need to be answered immediately. Therefore, emails and faxes create flexibility in the number of agents available for telephone traffic and they offer an effective instrument to control the corresponding service level. Notice that we can handle an infinite backlog of emails and faxes in the model, by adding a queue with infinitely many jobs. The objective can be to maximize throughput while meeting service-level constraints on calls. We will discuss in Section 3.6 a paper that considers this particular case. If agents are idle they have the possibility to handle these jobs. Of course, the arrival process of emails and faxes can also be modeled as, for example, a Poisson process, such that the queueing process behaves similarly to calls.

2.7 Planning of resources

Call centers aim to schedule the employees in such a way that objectives concerning costs and service levels are met. In the literature, a commonly used objective is to satisfy a constraint on a service-level measure and to minimize personnel costs. In this section we first list a number of properties that make planning a difficult problem. Next, we describe the features of the model with respect to planning.

The relation between the staffing levels and the total productivity of a call center fluctuates in practice. Often there is a deviation between the number of agents that is available according to the system and the number of scheduled agents. The reason is that agents are not fully productive during the whole day. Agents usually have several interruptions, for example, breaks and meetings. We learned from experience that the impact of these shrinkage factors can be huge; to meet service-level agreements, interruptions like breaks and meetings need definitely to be taken into account.

The shrinkage factors must be carefully analyzed when building and validating models. It is easy to see that the shrinkage percentage varies during the day. For example, if people take fewer breaks during busy hours, the shrinkage is lower. However, the impact of a change in shrinkage can be high

during periods in which much work arrives; a small decrease in productivity can easily result in much lower service levels.

Obtaining good schedules is a difficult problem in multi-skill contact centers. A reason is that the skill set of each individual agent influences the performance and needs to be taken into account while measuring performance, as shown by means of an example in Section 1.3. Moreover, bad workload predictions easily result in a mismatch between on the one hand the total working capacity available for a job type, and on the other hand the arriving workload of that type, yielding lower overall service levels. Thus, inaccuracy of predictions also contributes to the high complexity.

Also labor rules play a role in the planning of resources. They determine for example the maximum length of a shift and the minimum time between two consecutive shifts.

In our model, a shift is defined by a collection of working hours from $\mathcal{T}$ and a subset of skills from $\mathcal{M}$. We assume that for each shift there is a group of agents that has the skills to work that shift. Hence, for notational convenience we can denote the skill set of each shift with f_k, i.e., the index of the corresponding agent group for shift k, with $k \in \{1, \ldots, K\}$ and K denoting the total number of shifts. In order to meet the service-level constraints, we suppose that for every agent group there is a set of workable shifts such that for some agent configuration the requirements are met. In this context, a shift k is workable if there is a group g such that $f_k = g$.

A lower bound for the number of employees is the workload. The workload can easily be expressed in agent numbers if the service rates are class-dependent (and not group-dependent) $\sum_{m=1}^{M} \rho_m = \frac{\lambda_m}{\mu_m}$, with μ_m the service rate of type m and λ_m the arrival rate of type m. However, an inaccuracy is caused by the fact that the workload of a type can take real values, while agent numbers are integers. For example, a workload of 0.75 in case of four job types requires four specialists, while scheduling $4 \times 0.75 = 3$ generalists also suffices.

In case of group- and skill-dependent rates a lower bound can easily be calculated by $\sum_{m=1}^{M} \rho_m = \frac{\lambda_m}{\max \mu_{mg}}$. Still, this lower bound gives no guarantee on an integer solution close above it and, hence, systems can become unstable by using these solutions. The error due to the integer values can be even larger than in case of a single skill. To obtain an accurate lower bound, one needs to take the staffing levels of the groups into account. The next integer

linear program provides a more accurate lower bound

$$\min \sum_{g \in \mathcal{G}} y_g$$

subject to

$$\begin{array}{ll} Ax - Ey = 0, & \\ (Rx)_m \geq \lambda_m, & \forall m \in \mathcal{M}, \\ y_g \geq 0 \text{ and integer}, & \forall g \in \mathcal{G}, \\ x_{mg} \geq 0, & \forall m \in \mathcal{M}, \forall g \in \mathcal{G}, \end{array}$$

with E denoting the diagonal-identity matrix, y the vector with the staffing levels of each group, x_{mg} representing an activity, denoting the number of agents from group g that work on jobs from class $m \in S_g$. A column of matrix A takes the value 1 in row g if the activity associated with the column belonging to group g, otherwise it is equal to 0. A column of matrix R takes the value μ_{mg} in row m if the activity associated with the column is to serve jobs of type m, otherwise it is equal to 0. By adding

$$y_g \leq q_g^+ \text{ and } y_g \geq q_g^-, \quad \forall g \in \mathcal{G},$$

with q_g^- and q_g^+ the minimum and maximum size of group g, respectively, it is possible to control the sizes of the agent groups.

Moreover, it can be interesting to calculate staffing levels that minimize costs. To achieve this, we modify the objective function of the integer programming model by multiplying the staffing levels by costs, yielding $\sum c_g y_g$, with

c_g, the costs of staffing an agent in group g during one unit of time.

The parameter c_g is also used in Chapters 8 and 9.

Chapter 3

Literature Overview

This chapter gives a literature overview of routing, staffing, and scheduling in multi-skill call centers.

3.1 Introduction

This chapter is structured as follows. Section 3.2 gives practical examples of factors that complicate the analysis of a call center. In Section 3.3 we characterize the models from the literature. Section 3.4 elaborates on routing policies. In that context, a number of approximation techniques are listed that measure the performance in the multi-skill environment. Staffing is discussed in Section 3.5, where we focus on a multi-server queue and where we refer to papers with structural results. Theory about queueing systems with homogeneous servers is also relevant to multi-skill systems because it has been shown that under certain conditions the behavior of systems with homogeneous servers is almost similar to contact centers having fully cross-trained agents. In Section 3.6 we describe the literature about routing and staffing in the most elementary queueing systems, the so-called canonical designs. Finally, Section 3.7 discusses the most relevant papers concerning multiple skills in more depth, by means of treating different optimization methods.

The scientific literature pays increasingly attention to call centers. In Koole and Mandelbaum (2002) queueing models that are relevant to call centers are discussed. We refer to Gans, Koole, and Mandelbaum (2003) for a complete survey on mathematics in call centers. It contains a tutorial on how call centers operate, a survey of academic research, and an outline of

important problems that had not been addressed earlier.

3.2 Model features

This section discusses several factors that complicate the mathematical analysis of a call center. These factors are also useful because they denote the diversity of call centers. With regard to the scope, we restrict us to the factors that are not included in the model of Chapter 2. They are treated by summarizing the papers from which the results originate.

Brown, Gans, Mandelbaum, Sakov, Zeltyn, Zhao, and Haipeng (2005) characterize the arrival process at an inbound call center as a non-stationary Poisson process. We remark that even this is still an inaccurate description of the arrival processes in call centers. For example, redials (possibly caused by abandonments) can have a substantial impact on the results and the usefulness of methods. We refer to Aguir, Akşin, Karaesmen, and Dallery (2004) for a model with redials.

Discrepancy between models and reality is for example also present in service-time distributions. The literature most often assumes that service times are exponentially distributed variables, while Bolotin (1994) and Brown, Gans, Mandelbaum, Sakov, Zeltyn, Zhao, and Haipeng (2005) show lognormality in certain cases.

Jongbloed and Koole (2001) consider the arrival process as a Poisson process and explain that the arrival intensity is a stochastic variable itself.

Steckley, Henderson, and Mehrotra (2004) explain and analyze the stochastic relation between the prediction and the realization of the arrival process, for different time horizons. They relate the decisions about staffing levels to the real workload by means of service-level measures.

Whitt (2004) and Sisselman and Whitt (2004) try to improve the routing such that the satisfaction of the customers and agents increases by paying attention to the working circumstances of the agents. They write: “skill-based routing is designed to ensure that calls are not only handled promptly but also resolved properly”. Their solution is an extension to the priority routing from Wallace and Whitt (2005) because the preferences of agents, with respect to their skills and schedule, are taken into account as well.

Gans and Zhou (2004) consider an outsourcing model with high- and low-priority customers. High-priority customers should get a high service level, while the throughput of low-priority customers is maximized. The decision

is to accept or reject low-priority customers. The rejected customers are outsourced.

Whitt (2006) models the number of available agents as a stochastic variable, because of absenteeism.

3.3 Types of models

While studying the literature we observed that routing and staffing issues are most often analyzed separately. This section addresses the main characteristics of the corresponding models from the literature.

The literature treats at least three different settings of job routing, namely:

- call routing in multi-skill inbound call centers,
- call blending with faxes and emails in contact centers, and
- call routing in single-skill call centers with calls having different priorities.

We note that from a mathematical point of view the different routing problems have much in common. Many models are a special case of a multi-skill inbound call center with different priorities associated with each skill. This explains the importance of research on routing in multi-skill call centers and that results can be useful for solving different problems.

Models that support the routing and staffing in multi-skill call centers have in common that agents are grouped. Agents from the same group are assumed to behave statistically identically. In conjunction with agent groups, the assignment of jobs to employees often occurs according to priority routing policies, see Section 3.4.2.

3.4 Routing policies

This section treats some important subjects related to routing policies. In the remaining sections a number of papers from the literature are discussed. In case of differences with the model of Chapter 2, a short discussion is added.

Push and pull systems

There are several ways to implement call-assignment policies. We make a distinction between push and pull systems.

- In a push system an arriving call is assigned to an available agent that is chosen by the computer system. That agent is responsible for the service of the customer.
- In pull systems different agents are simultaneously notified about a call in the queue. The agent that reacts fastest handles the call.

Pull systems give agents more flexibility, which is an advantage. On the other hand, from the perspective of an agent, it might be attractive to ignore a call, and for example have a break instead. We note that in the literature it is often assumed that optimal control of the manager involves push systems.

3.4.1 Standard solution approach

An approach to analyze call centers is to apply the theory of Markov decision process, see Chapter 4 for details. When modeling a call center as a Markov decision process it is theoretically possible to take all relevant information (that is available at the decision epoch) into account, as long as the state is memoryless, for example the hour of the day, the queue lengths and the productivity of each individual agent. The information required to describe future events of the process is called the state. With respect to the analysis it is important to keep the state as small as possible. The reason is that standard analyzing techniques calculate a function for all possible states of the system, i.e., the state space. This can be a huge number of calculations. The fact that the number of states grows exponentially in the dimension of the problem is called the curse of dimensionality, see Bellman (1961).

In a Markov decision process the system moves from one state to another. This is called a transition. In most models from the literature the state changes when an arrival occurs or when the service of a job finishes. These are the two common epochs at which it is decided if an agent starts serving a call. At arrival, calls are either queued or immediately assigned to an agent, depending on the routing policy. When multiple agents are available, the policy determines the agent that serves the job. Hence, the policies used at arrival epochs are also called agent-selection policies. At service completions, agents become available again. A call-assignment policy prescribes if an

agent becomes idle or starts to serve another job. Relevant methods are discussed in Section 3.7, in the paragraph about call selection.

3.4.2 Types of policies

Different types of routing policies exist. Dynamic routing is discussed first, overflow routing or priority routing is explained next, and finally hierarchical routing is treated.

Dynamic routing

Dynamic routing policies are policies that take the state of the system into account.

Optimal dynamic routing policies can be calculated by using methods from dynamic programming, which is part of the theory on Markov decision processes. We make a distinction between two types of applications of dynamic programming. In the first place, it is used in the literature for the derivation of structural results. Three examples are given:

- Hordijk and Koole (1993) study scheduling problems of multi-class customers on identical processors. Jobs arrive according to a controllable Markov process. As a special case they show the optimality of the μc-rule in the last node of a controlled tandem network.

- The two papers that we mentioned about call blending, Bhulai and Koole (2003) and Gans and Zhou (2003), are worth mentioning again, see Section 1.1.

- Carr and Duenyas (2000) analyze job admission to a single-server queue with two job types, see Section 3.6.

- In Koole and Pot (2006a) dynamic programming is used to determine the optimal number of lines and agents in profit centers.

In the second place, dynamic programming is useful to analyze models numerically. An example of an application is the calculation of the steady-state distribution of the Erlang-C model. However, optimal routing policies are difficult to obtain for multi-skill call centers. The associated problems are hard to solve due to the huge number of states. Several experiments show that only single-skill call centers and small multi-skill call centers are

tractable. Notice that these models are already simplified as much as possible, such that the state contains a minimal amount of information. This is, for example, achieved by grouping agents together by assuming that agents with the same set of skills behave statistically the same. This reduces the size of the state space because no distinction between agents from the same group is required. In the literature a number of other simplifications are usually made, e.g., exponentially distributed service times, and a stationary arrival process, see for example Bhulai and Koole (2003) and Koole and Pot (2005).

Priority or overflow routing

We characterize priority routing or overflow routing as follows, see for example Wallace and Whitt (2005). Assume that jobs from different types arrive according to independent arrival processes and agent groups consist of agents with equal skill sets. An overflow routing policy for agent selection specifies for each call type an ordering of the agent groups. The ordering is chosen in such a way that jobs are assigned to the first available agent, while traversing the different groups according to the ordering. Thus, the group with the highest priority is considered first and if an agent is available the job is assigned to the agent, otherwise the group with the second-highest priority is considered etc. It is possible to visualize this in a network, with each node representing an agent group and the arcs denoting the ordering of the priorities for each call type. A job is assigned to the first available agent in the network and is blocked if the agents from the groups along the path are all busy. Jobs leave the network immediately after service completion.

Next to agent-selection policies, call-assignment policies can be specified in a similar way. In that case, priorities are used to determine for each group the type of job that is served at a service completion. If a job from the class with the highest priority is available, it is served, otherwise the queue that contains the jobs with the second-highest priority is considered, etc.

We discuss agent-selection policies in further detail. A hierarchical agent-selection policy is defined by an $N \times M$ matrix, which we denote by π. Each row specifies an agent group and each column a job type or skill. Column m specifies the priority of each group $g \in \mathcal{G}$, associated with skill or job type m. Value 1 indicates the highest priority and value k indicates the group with the k-highest priority. Value 0 indicates that the group does not have

skill m. For example,

$$\pi = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 2 & 1 \\ 2 & 3 & 2 \end{bmatrix}. \tag{3.1}$$

This matrix contains the following information. Type 1 is handled by groups 1 and 3, and group 1 has a higher priority than group 3. Type 2 is served by all three groups and the priority increases with the group index. Finally, jobs of type 3 can be served by agents from groups 2 and 3, and group 2 has a higher priority than group 3.

Sisselman and Whitt (2004) consider priority-routing policies. They assume that there are a sufficient number of agents (with the right skills) available to satisfy the service-level conditions. They present an algorithm for setting up routing policies that support skill preferences of agents, for example, by giving sufficient idle time to agents. This is achieved by including different control parameters in the model. An advantage of the model is on the one hand satisfaction of both customers and agents and on the other hand high flexibility by means of the control parameters. However, the control parameters make the implementation time-consuming and, additionally, the control parameters might also influence the computation times. A limitation is that the framework is not capable of identifying an optimal routing algorithm, as they mention in their paper. Service levels can only be measured by simulation.

Hierarchical routing

Hierarchical routing is a special type of overflow routing, see for example Franx, Koole, and Pot (2006). Like overflow routing, the priorities of the groups for each type can be described by a matrix, with a column representing a job type and the m-th column containing the priorities of the groups for the particular type, see for example Equation (3.1). The difference with overflow routing is that an ordering of group priorities is required among the different job types. With hierarchical overflow routing there exists an ordering of the skill types such that $\pi_{rj} > \pi_{sj}$ and $r > s$. The example from the previous section satisfies this condition.

Routing policies and staffing levels are related to each other. With hierarchical routing, it is possible that the service level increases when an agent with two skills is replaced by an agent with one skill. This changes the staffing levels of two groups, while the total number of agents remains the

same. However, when using optimal dynamic routing policies, it is easy to show that the service level would definitely decrease. We refer to Section 1.3 for this example.

With respect to the storage in computer systems, hierarchical and overflow routing are among the simplest kinds of routing policies to describe and to save. These policies can be specified by a (relatively) small matrix and it is not necessary to store actions for every state of the system, in contrast to dynamic routing policies. Hence, the configuration of hardware is easy and the policy requires little memory.

Section 3.4.3 gives a literature overview of approximation methods from the literature for blocking models with overflow routing.

3.4.3 Approximation techniques

Due to the high mathematical complexity of analyzing multi-skill call centers, approximation methods can be useful. The relevant papers are discussed next.

Conditioning

In Shumsky (2003) the performance of a call center with two types of customers is approximated. The state space is partitioned into several regions and the conditional system performance within each region is approximated. This yields a substantial reduction in computation times. Moreover, errors are most often smaller than ten percent and on average less than five percent.

Overflow routing in blocking models

We refer to Section 3.4.2 for a description of overflow routing. Recall that with overflow routing, the agent groups are considered one after another according to the priorities. If all agents in a group are busy, the job 'flows over' to the next group. In this section we consider blocking systems. If there is on arrival no agent available with the right skill, the call is blocked.

In blocking systems it is straightforward to express the service level as a function of the blocking probability. A great advantage of overflow routing in blocking systems is that fast approximation methods for calculating the blocking probabilities have been developed. These are discussed next.

The Equivalent Random Method is a well-known method in the area of computer networks and tele-communication. A description is given in

Cooper (1981), pages 165–171. The method is derived from Kosten (1973), in which a formula is presented for the peakedness of the overflow process of an $M/M/s/s$ system. This formula has been developed by Wilkinson (1956) and Bretschneider (1956). Several other people contributed to the usefulness of the method by developing numerical approximation methods for the inverse of this formula. The method is generalized to call centers in Tabordon (2002).

Hayward and Fredericks extended the work that was presented in Kosten (1973), see Fredericks (1980) and Wolff (1989) (pages 354–355), and developed the Hayward-Fredericks method. Their contribution is an approximation method for the overflow process of a $G/M/s/s$ system, where the arrival process is determined by the first moment and the peakedness factor. The resulting method decomposes a multi-skill blocking system with overflow routing into separate multi-server groups. The method is extended to call centers in Chevalier and Tabordon (2003).

The Interrupted-Poisson-Process method is developed by Van Muylder. A description can be found in Tabordon (2002). The fundamental idea is that the superposition of different overflow streams is assumed to be a renewal process. Based on this assumption, an approximation is found for the Laplace transform of the inter-overflow times.

The Poisson method is developed by Koole and Talim, see Koole and Talim (2000). The overflow processes are approximated by Poisson processes. As a result, the superposition of overflow streams are approximated by the same type of process. This method provides an upper bound of the blocking probability, because the burstiness of the streams is underestimated. The impact of underestimating the burstiness is significantly present in systems with a low workload.

The HyperExponential-Decomposition method is described in Franx, Koole, and Pot (2006). It decomposes the routing network into separate agent groups. The overflow processes are approximated by processes with second-order hyperexponentially distributed inter-arrival times. An exact analysis is applied to each group. The authors report that this method yields the most accurate results compared to the other methods.

Approximate dynamic programming

Koole and Pot (2005) apply approximate dynamic programming, i.e., the value function is approximated by fitting a simple structure into the dynamic

programming equations. In Bhulai (2005) a dynamic policy is derived by using the value function of a simpler queueing system. We refer to Section 9.1 for more details about the experiments of both papers.

3.5 Staffing and limiting regimes

This section discusses the literature on staffing and, and especially results obtained by considering limiting regimes. We note that most of the papers from the literature using limiting regimes are concerned with a single group consisting of homogeneous servers. The papers relevant to systems with multiple skills and agent groups are discussed in Section 3.7.

Classifications of staffing

Staffing is a broad subject and therefore several classifications are possible. We describe two classifications, related to: the time horizon and the level of staffing.

Time horizon: With respect to the time horizon of decisions, a proper classification of planning is: strategic, tactical, and operational. These are discussed next. An example of a strategic problem is starting an additional call center in a different country to decrease personnel costs. An example of a tactical problem is the hiring of new employees, the choice about their skills, and the career path of the current employees, including training, see Gans and Zhou (2002). An operational problem is, for example, the scheduling of employees for the next period (in terms of weeks or months). Examples of short-term operational problems are the determination of the kind of jobs that the employees work on, their skill set, the routing policy of the different activities to the employees, and the use of agents with flexible contracts. Most of the references from this thesis are related to this subject.

Level of staffing: A second classification is the level of expressing staffing levels:

- agent groups (with each agent having the same set of skills),
- teams that contain agents with different skill sets, and
- the call center as a whole.

The same classification is possible with respect to leadership levels. The names of the corresponding managers are usually called group, team and call center manager, respectively.

In software packages, elementary queueing models like Erlang C (see for example Cooper (1981)) are often used to determine staffing levels. These models assume that all agents behave the same, such that the skills of the agents are disregarded. Our experience is that, for more detailed studies that take routing policies and the skills of the agents into account, simulation is frequently used by the industry.

Optimal staffing is discussed in Akşin and Harker (2003). They consider a model with parallel independent multi-server queues and one shared information system. All servers that handle calls need to communicate with this system. This occurs simultaneously and is modeled as a processor-sharing service discipline. They obtain product form solutions and expressions for performance measures. The model does not fit in our framework from Chapter 2, because of the shared information system.

Limiting regimes

An important finding with respect to staffing is the square-root safety staffing rule, see Borst, Mandelbaum, and Reiman (2004) and Green and Kolesar (1991). This rule relates changes in the offered workload to the required number of agents s such that the service level remains equal. This rule is given by

$$s = a + \beta\sqrt{a},$$

where $a = \frac{\lambda}{\mu}$ is the offered load (λ=arrival rate, μ=service rate) and β represents the service level. The square-root safety staffing rule has a wide range of applications. Several illustrations are given in the papers mentioned below.

The Halfin-Whitt regime or Quality and Efficiency Driven (QED) regime, introduced in Halfin and Whitt (1981), is an example of a limiting regime. Under this regime the number of servers goes to infinity and the arrival rate is appropriately scaled such that the service-level remains constant (in the limit). This regime justifies the square-root safety staffing rule. In Whitt (1992b) a multi-server queue is analyzed. On the one hand the relation between workload and server utilization is considered, and on the other hand the square-root safety staffing principle is discussed. The second one is extended to general arrival and service time distributions. Moreover, several

implications are listed. Borst, Mandelbaum, and Reiman (2004) present an overview of the different limiting regimes and relate them to each other. In addition, the square-root safety staffing principle is revisited and extended with costs for staffing and delay.

A single skill: In Whitt (2006), a multi-server group is considered and approximated by analyzing the corresponding fluid model. The model and the analysis allow general distributions for the arrival process, the service times, and the abandonment times. The main objective is to determine the optimal number of agents that maximizes a profit function composed of revenue for throughput and costs concerning the number of servers in use, abandonments and waiting time. Under certain conditions (non-decreasing and convex cost functions and non-decreasing concave revenue functions) a solution is found. The effectiveness is extensively validated for different inter-arrival time distributions.

Multiple skills: Reiman (2000) is relevant to the multi-skill setting, in which diffusion limits are considered.

In Harrison and Zeevi (2005) and Bassamboo, Harrison, and Zeevi (2004) the arrival rate is scaled up by a super-linear function and the service and abandonment rates grow linearly. These papers are discussed in Section 5.3.

The conventional heavy-traffic limit theory is applied to a call center in Harrison and Lopez (1999). It is the first paper in which a dynamic control problem is explicitly solved in the multi-pool multi-class setting using the conventional theory of heavy-traffic limits. The routing policies are modeled by assuming that the system is in a heavy-traffic situation. In this regime the arrival rates and service rates are accelerated linearly such that the system utilization approaches one. Next, asymptotically optimal policies are derived. It is explained that these policies can be useful in controlling the original system. The policies suggest to hold backlog in one particular job class, and to utilize the servers fully unless the total backlog is smaller than a certain threshold.

3.6 Canonical designs

Insight is often obtained from simple examples. In the context of contact centers these are called the canonical designs and will be discussed next.

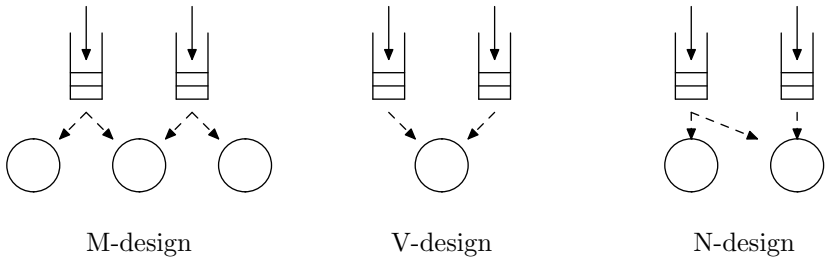


Figure 3.1: Canonical designs

We discuss the V-design, N-design, and M-design. They are useful to get a better understanding of routing.

Some designs are not discussed in this section. The I-design represents a single-skill call center and is for that reason relevant to staffing, see Section 3.5 for a literature overview. The Λ-design classifies single-skill multi-server queues. If the servers are not homogeneous this design is concerned with agent selection policies, see Section 3.7. We consider the W-design in combination with the M-design as a multi-skill multi-server system, as described in Chapter 2. That chapter gives a general description of a multi-skill call center, while the relevant literature is discussed in Section 3.7.

V-design

With V-design one refers to a model with two job classes and one group of homogeneous servers, as drawn in Figure 3.1. Perry and Nilsson (1992) consider a system with two classes of calls that are served by a single pool of agents. They determine the required number of agents and an assignment policy to satisfy a target on the expected waiting times.

We discuss two papers on call blending. The models are the same and treat the case with two types of jobs: inbound calls and outbound jobs. The inbound calls arrive according to a Poisson process. The number of outbound jobs is unlimited and the jobs represent the backlog. (Unserved calls wait in a buffer until they are served.) The objective is to maximize the number of finished outbound jobs. There is a constraint on the average waiting time of inbound jobs. The question is how to schedule an available agent. The choices are: keep the agent idle, assign an inbound job, or assign an outbound job.

In Bhulai and Koole (2003), the case with equal service rates for the inbound and blending work is solved to optimality, and the structure of the optimal policy is characterized. The starting point of their analysis is the value function, well known from dynamic programming. In contrast, Gans and Zhou (2003) analyze the model using a linear programming formulation. In addition, they obtain results for the case with unequal service rates. The optimal policy is a threshold-reservation policy.

Armony and Maglaras (2004b) and Armony and Maglaras (2004a) analyze a multi-server call center with two types of customers. At arrival, customers receive information about their expected waiting time and can choose between hanging up, waiting until a server is available (the first type), and a call-back service (the second type). Customers get a guarantee on the maximum delay before receiving a reply. They propose a nearly optimal policy that serves customers from the second type when the queue length exceeds a threshold. The paper also gives an approximation of the performance for different measures. It shows that simple routing schemes can be very effective and can lead to models that are easy to solve. The call-back service is not included in the model from Chapter 2 because, as far as we know, call-back options are not a standard feature of multi-skill contact centers and analyzing the current model of Chapter 2 is already complicated enough.

Brandt and Brandt (1999) is to some extent related to call blending. It assumes two job types with equal service rates. High-priority jobs arrive by the live customers. If they decide to leave the queue due to impatience, a low-priority job is created that represents the call-back message. This paper develops performance measures for a fixed threshold policy.

Van Mieghem (1995) proves asymptotic optimality of a simple generalized $c\mu$ rule with waiting costs that are convex and increasing. This result is generalized to a system with job-type dependent and server-dependent rates in Mandelbaum and Stolyar (2004).

The literature on tele-communication contributes substantially to our knowledge of contact centers. Blanc, De Waal, Nain, and Towsley (1992) consider a multi-server queue with two job types, different rewards per type and a first-come-first-serve service discipline. The system is controlled by the admission of the job type with the lowest reward. The objective is to maximize the total discounted rewards. The optimal policy admits the jobs with the lowest reward only if the total number of jobs is below a bound, the threshold. Admission is not considered in Chapter 2. Call centers usually serve all customers.

Guérin (1998) presents a model without queues. It contains a multi-server group, which receives low- and high-priority jobs. He develops an admission policy for the low-priority jobs such that the fraction of blocked high-priority jobs is bounded, and they analyze the system under that policy.

In the context of retail, Berman and Larson (2000) consider a two type system. High-priority customers arrive to cash registers according to a stochastic process, and generate workload. Meanwhile there is a quantity of low-priority work to be handled by the servers, such as restocking and maintenance activities. A switching cost is included for each time that a server changes from high-priority jobs to the low-priority work. The paper describes two heuristics to control the servers. We remark that switching costs and times are also realistic in call centers because productivity decreases when cross-trained agents switch between jobs of different types. However, it is for simplicity not included in the model of Chapter 2. Moreover, switching times can be modeled implicitly by decreasing the service rates.

Schaack and Larson (1986) use generating functions to characterize the performance of threshold reservation policies for systems with $M \geq 2$ classes of customers, all with the same exponential service time distribution. The paper concentrates on performance analysis and does not include a notion of optimality as it analyzes policies. Furthermore, it does not address service-level constraints explicitly.

Carr and Duenyas (2000) consider a single-server queue with two job types and different service standards. They discuss a job admission and sequencing problem.

Örmeci, Burnetas, and Emmons (2002) consider dynamic admission control in a loss queueing system with two classes of jobs with different service rates and random revenues.

Peköz (2002) considers a multi-server non-preemptive queue with high- and low-priority customers. The decision maker decides when waiting customers enter service. The goal is to minimize the mean waiting time for high-priority customers while keeping the queue stable. An asymptotically optimal policy is derived using a linear programming approach.

N-design

Stanford and Grassman (1998) consider a model with two skill types and two agent groups: one group of specialists and one group of generalists, as depicted in Figure 3.1. Jobs are served according to priority policies that

are constant over time. Shumsky (2003) proposes an approximate analysis, see Section 3.4.3. Bell and Williams (2001) prove the asymptotic optimality of threshold controls in the conventional heavy-traffic limit.

M-design

The M-design, see Figure 3.1, concerns a model with two groups of specialists and one group of generalists having both skills. There are no waiting rooms ($L_m = 0$).

The model is analyzed in Örmeci (2004), under the assumption that the specialists work faster than the generalists. The paper shows that it is optimal to give the specialists a higher priority than the generalists. Concerning the generalists, a sufficient condition is derived under which it is optimal to accept jobs if no specialist is available and a generalist is available.

3.7 Optimization

This section is devoted to optimization algorithms for multi-skill contact centers. We elaborate on a number of papers that contribute to our understanding of multi-skill contact centers and especially of routing. The papers help to answer the following questions: How to improve routing policies and plan agents, and, given a policy, how to determine the service level?

3.7.1 Shift scheduling

In this section we address the most important papers about shift scheduling in single-skill call centers and we describe two papers concerning multi-skill call centers. The different models and methods have in common that a period of one working day is considered.

Single-skill

With respect to single-skill call centers we mention two papers that are in our opinion the most interesting ones. The most elementary integer linear

programming model originates from Dantzig (1954)

$$\min \sum_{k \in \mathcal{K}} c_k x_k$$

subject to

$$\begin{aligned} \sum_{k \in \mathcal{K}} a_{k,t} x_k &\geq s_t, && \forall t \in \mathcal{T}, \\ x_k &\geq 0 \text{ and integer}, && \forall k \in \mathcal{K}, \end{aligned}$$

with c_k the price of shift k, s_t the staffing level in period t, and $a_{k,t}$ denotes 1 if shift k is active during interval t, and 0 otherwise. The decision variable x_k indicates the number of times that shift type k occurs.

Thompson (1997) considers two types of service levels: aggregate threshold service levels and minimum acceptable service levels. A method is introduced that takes both types of service-level constraints into account. The method integrates the determination of staffing levels and shift scheduling, and the problem is solved by using linear programming.

Atlason, Epelman, and Henderson (2002) and Atlason, Epelman, and Henderson (2004a) consider optimal shift scheduling in single-skill call centers under the same types of service-level constraints. They propose a general methodology based on the cutting plane method of Kelly Jr. (1960) that integrates the determination of on the one hand staffing levels, and the other hand shifts. They assume that the service level cannot be easily computed and instead, is evaluated using simulation. The problem is formulated as an integer programming problem in conjunction with the service-level constraints, which are included via non-linear constraints. They relax the problem from the non-linear constraints and use standard methods to solve the remaining set-partitioning problem. In case service-level constraints are violated the cutting-plane technique is applied. First, the violated constraints are detected and next cuts are added such that

- the current solution is removed from the feasible region and
- no optimal solution is lost.

This continues iteratively until a feasible solution is found. The algorithm produces good results, but the computation times are considerable due to the simulations.

Multi-skill

Cezik and L'Ecuyer (2005) extended the method of Atlason *et al.* to the equivalent problem in multi-skill call centers. Due to the higher complexity as opposed to single-skill call centers, the computation times are longer. Consequently, the shift-scheduling problem is not tractable numerically, such that they consider shifts with a duration that is equal to the opening time of the call center.

Bhulai, Koole, and Pot (2006) propose a two-step method to generate shifts for multi-skill call centers, see Chapter 9 for the details. In advance, the day is split into different consecutive time intervals, typically half an hour or one hour. In the first step, the optimal staffing level for each group and each interval is determined. In the second step, shifts are composed such that the staffing level in each interval is met. This approach reduces the required number of simulations. An example is discussed that is solved close to optimality.

3.7.2 Staffing

The papers that mainly contribute to our understanding of staffing in multi-skill call centers, are treated next.

Loss model

Chevalier, Shumsky, and Tabordon (2004) search for a compromise between service level and staffing budget. Fast optimization is achieved by looking at 'equivalent' blocking systems and using an efficient approximation algorithm, discussed in Section 3.4.3. Their model consists of an overflow routing scheme with several groups of specialists and one group of fully cross-trained agents. The latter group is added for flexibility. Their conclusion is to spend eighty percent of the staffing budget on specialists and twenty percent on cross-trained agents. The ratio between specialists and flexible agents in a call center with two skills has been analyzed before in Shumsky (2003).

Credit schemes and characterization of feasible agent configurations

Borst and Seri (2000) consider the model for inbound multi-skill call centers as we described in Chapter 2: Poisson arrivals, different job types, exponen-

tially distributed service times, etc. The service level is measured by the average waiting time. To ensure that a desired level of service is reached, a set of sufficient conditions and a set of necessary conditions on the number of agents are derived. In order to derive these conditions, two situations are considered. First, a call center with only specialists is analyzed. Second, they look at the case that all agents operate as generalists. The results from these two situations are applied to the case in which agents have different sets of skills. In addition, two routing schemes are introduced that prescribe the call selection decision. They ensure that the service level is similar to, or better than, one would have experienced in the first situation.

LP with recourse

In Harrison and Zeevi (2005) the total personnel costs and the expected abandonment costs are minimized, taking into account the skills of the agents (per group) and the routing strategy. The number of agent groups and customer classes is given beforehand. The arrival rates are allowed to be time-dependent and to vary stochastically. The procedure is as follows. In the first stage, the system is considered as a regime that scales up the service and abandonment rates linearly and the arrival rate more than linearly; it is the so-called local fluid version of the dynamic scheduling problem. The optimal staffing vector is determined by an optimization procedure. During the optimization, exactly one skill is assigned to each agent. The expected abandonment costs are approximated using fluid limits. In the second stage, the capacity vector is optimized by minimizing the associated personnel costs and costs that represent the system performance. The model from the first stage is used for a fixed capacity vector. This is a so-called two-stage LP with recourse according to the literature about stochastic programming.

Bassamboo, Harrison, and Zeevi (2004) consider the same model. Under a limiting parameter regime they give a lower bound on the expected total costs. In addition, they propose a method for staffing and routing that achieves this lower bound.

Pooling

We treat pooling as a synonym of merging. In Tekin, Hopp, and Van Oyen (2004), methods are presented to measure the effect of pooling different departments. They assume that the departments are mutually independent, such that each department can be modeled as a multi-server queue. The

impact of pooling is analyzed by using approximations of the multi-server queue. They consider general service-time distributions, different department sizes, and generally distributed inter-arrival times, which differ among the departments. The underlying pooling models are described in Kleinrock (1975). In comparison to the model of Chapter 2, general service times are not modeled in that chapter. However, departments do fit in the model of Chapter 2 because a department is technically similar to an agent group. Smith and Whitt (1981) and Benjaafar (1995) show that a pooled system is better than a dedicated one if arrivals and service times have the same distribution. In Van Dijk (2002) is shown that pooling is not necessarily an improvement, in particular if the service distributions are different per class. Moreover, much literature is available on pooling in serial service environments.

3.7.3 Routing

The following papers treat the optimization of routing policies in multi-skill call centers.

Simulation procedure

Wallace and Whitt (2005) consider priority routing. A procedure for optimizing staffing and call routing is proposed, which is achieved by using limited cross-training. Several types of performance measures are included. We give a brief description of the algorithm: The total number of agents is determined by assuming that all agents have all skills. Then first all agents are replaced by specialists, and next additional skills are assigned to the agents. Finally, the staffing levels are optimized by means of simulation. In addition, they show numerically (by means of simulation) that when each agent has at most two skills, the performance can be almost as good as when each agent has all skills, which argues the use of the classical Erlang models. Thus, by adding a little flexibility for routing, it is possible to minimize staffing levels. This holds even in conjunction with simple static routing policies.

However, the result is only shown for call centers with equal mean service times for each type of job and no holding costs. The question is whether these assumptions are realistic and whether the results would also hold otherwise. We conjecture that they do hold, because, by assigning two skills to each

agent, the number of agents available with a certain skill can double. This is expected to have considerable impact.

Overflow routing

This section presents a number of methods that are helpful to improve and optimize service levels. The methods based on approximations of blocking probabilities, see Section 3.4.3. For a description of overflow routing we refer to Section 3.4. One could expect that blocking models are not useful to call centers because call centers have queues. However, they appear to be useful with respect to optimization. The methods compare the blocking probabilities of call centers that undergo small adjustments. For example, when the staffing level of a certain agent group is increased by the integer value 1. It appears that the differences in blocking probability give a good indication of the performance change of similar systems that include queues. This kind of methods are given in Koole, Pot, and Talim (2003), Chevalier, Shumsky, and Tabordon (2004), and Chevalier and Van den Schrieck (2005). The studies show that the optimized instances are nearly optimal, also in the corresponding delay systems. The second reference has been discussed in Section 3.7.2. The other two papers are described next.

In Koole, Pot, and Talim (2003), a local search method is proposed. The objective is twofold: low staffing levels and good routing policies under service-level constraints. As already explained, both issues are related to each other. They demonstrate that even with simple optimization procedures good routing policies can be obtained.

In Chevalier and Van den Schrieck (2005) an optimization method is applied to determine the optimal staffing level of each agent group. They combine a Branch and Bound algorithm with the Hayward method. The method is effective and performs well.

Approximate dynamic routing

In Koole and Pot (2005) approximate dynamic programming is applied to a call center with several groups of specialists and one group of generalists, see Chapter 5 for a more extensive discussion. A different holding cost is associated with each job type, representing the priority of the job type. The objective is to minimize the average holding costs of the calls in the queues. Using Little's law, this is equivalent to minimizing the weighted

average waiting time. It is shown that approximate dynamic programming works well for instances with two or three skills. Examples with up to fifty agents are provided and the performance of the policies is about five from optimality.

In Bhulai (2005), a heuristic method is proposed to assign calls to agents in blocking systems, which behaves well in large call centers. The policies are nearly optimal.

Call selection

If an agent completes a job, the system assigns a new job to the agent. We are interested in good rules to determine the next job. Relevant literature consists of the papers discussing queueing systems with priority routing. Although, these papers treat the single-server queue, these rules are also applicable to multi-server systems, such as call centers.

Fixed priority (FP) routing is the easiest policy to implement and to analyze. In case of general service time distributions, and non-preemptive and preemptive service disciplines we refer to Kleinrock (1975). A disadvantage is the limited possibility to control the first two moments of the waiting-time distribution.

The shortest remaining processing time (SRPT) policy minimizes the average waiting time, see Schrage and Miller (1966). According to their opinion, the non-preemptive case is hard to analyze. We remark that higher moments of the waiting-time distributions are not analyzed.

The class of time function scheduling (TFS) offers a more general framework for routing policies (which includes FP as a special case). These policies schedule customers according to general functions of the time customers have spent in the system. Most of the literature concerns TFS with linear functions of the time in system. Kleinrock (1975) discusses TFS with exponential service time distributions. He presents a formula for the average waiting time in case of preemptive and non-preemptive scheduling. Lu and Squillante (2004) consider the TFS with general service time distributions under a non-preemptive service discipline. TFS policies can be fair in many situations and they can also be a mixture between first-come-first-serve and smallest jobs first. They give closed-form expressions for the first and second moment of the waiting-time distribution. Furthermore, they show that it is possible to control the first two moments very precisely.

For the multi-class single-server case (a special type of V-design) it is

shown in Federgruen and Groenevelt (1988) and discussed in Walrand (1988), that the μc-rule is optimal among the work-conserving policies. The higher the value of $\mu_m c_m$ the higher the priority of class m.

Yahalom and Mandelbaum (2005) consider the multi-class multi-server queue with Poisson arrivals (V-design) and equal service rates. The objective is to minimize the discounted holding costs in the long run. All servers are equal and there is a fixed holding cost c_m for each job type, with $m \in \{1, 2, \ldots, M\}$. They show that the optimal policy is a combination of the μc-rule with a threshold policy on the number of reserved agents $K_m(x)$ for each type m, depending on state x. At a service completion, the next job is of type m if there are no customers of a type with a higher priority than type m waiting in one of the queues, and there are more than $K_m(x)$ idle servers. The problem of choosing the appropriate threshold levels is not considered. Notice that the policy is not work-conserving.

Chapter 4

Process Description and Standard Solution Methods

In this chapter we discuss standard techniques from the literature to analyze multi-skill call centers. We are in particular interested in measuring performance and obtaining optimal routing policies. It is shown that the complexity is too high to deal with call centers of a realistic size. This justifies the development of approximation methods and heuristics, which is the subject of later chapters.

4.1 Introduction

This chapter is structured as follows. In Section 4.2 we formulate a Markov decision process for the model of Chapter 2, see Puterman (1994) for an introduction and for additional information about the content of this chapter. While the queueing process of a multi-skill contact center evolves in continuous time, it is transformed to discrete time, making it possible to consider the system only at time epochs at which transitions occur. Examples of transitions are arrivals, service completions, and abandonments. As a result, several techniques from the literature become available.

The theory of Markov decision processes offers techniques that are generally applicable to optimize system performance. These techniques are the subject of Section 4.3. They are in theory useful for many different purposes, e.g., analysis and optimization. However, with regard to contact centers the practical value is limited; computations are intractable because of the curse of dimensionality and the difficult structure of the transitions complicates

the derivation of structural results. Both aspects are explained and illustrated in Section 4.4, by giving a precise description of the process and by means of numerical examples, respectively. In Section 4.5 we mention analytical techniques to obtain structural results for multi-skill call centers. One of them is applied to a few proofs of theorems that are relevant to the subject of this thesis.

4.2 Process description

In Chapter 2 we mentioned the most essential parts required to model a contact center as a queueing process. In this section the description is completed by formulating the system as a Markov process, see Section 3.4.1. An important element of a Markov process is the state. The state includes all relevant information to describe the system at an arbitrary moment in time, such that the next state only depends on the current state and is independent of the previous states. Other important elements are the transitions between the states. Transitions represent events that occur in the system and that change the system over time.

The notation in this chapter is almost the same as in Chapter 2. The only difference is that the superscript t is omitted in the model parameters in this chapter, because we study stationary behavior. Instead, a variable t is used. This variable has a different meaning and is not used as superscript.

Features

In summary, we consider a call center with M job types and G agent groups. In this chapter we consider two types of events: job arrivals and service completions. For simplicity and improvement of the readability we ignore abandonments in this chapter (except for the next section). In reality, at each event a decision is made about the routing of the jobs present in the system, which possibilities are denoted by actions. The events and actions from the real system are modeled by transitions between states. Typically, in our model the randomness of the process is due to the fact that events occur at random moments in time. However, the policies that we consider are not random themselves. Given an event, the policy determines the next state with a probability of 1. Moreover, we consider non-preemptive policies.

We define the state x as the number of busy agents in each group, and

the lengths of the queues. Thus, x is a vector

$$x = \left(x_1^q, x_2^q, \ldots, x_{M-1}^q, x_M^q, x_{1,1}, x_{2,1}, \ldots, x_{M-1,1}, x_{M,1}, x_{1,2}, \ldots\right),$$

with x_{mg} denoting the number of agents in group g serving jobs of type m, and x_m^q denoting the number of jobs in the queue of type m. We add a comma to the subscript of x_{mg} to avoid ambiguity. The state space contains finitely many states, because the queue sizes are limited, and is denoted by $\mathcal{X}$. More information about the state space will be given later in this section.

Uniformization

The notion of transition rates is only valid in continuous time. Hence, we apply uniformization to transform the system to discrete time, see Lippman (1975). We divide all transition rates from Chapter 2 by α such that the sum of all transition rates in each state is smaller than 1 and sum up to 1. The outcomes can be considered as transition probabilities in discrete time, and are denoted by a tilde. For example, $\tilde{\lambda}_m$ is the probability of an arrival of type m. A dummy transition is added to every state such that the total probability of leaving a state sums up to 1. In this way, there is no difference in observation between the system behaving in discrete time and the original system observed at exponentially distributed times with parameter α. It is sufficient to choose

$$\alpha = \sum_{m \in \mathcal{M}} \lambda_m + \sum_{g \in \mathcal{G}} s_g \max_{m \in \mathcal{M}} \{\mu_{mg}\} + \sum_{m \in \mathcal{M}} L_m \gamma_m,$$

which is an upper bound of the highest possible transition rate of leaving a state.

Control policy

The performance of our contact center is controlled by a policy. A control policy defines an action for each state. The transition probabilities between two states depend on the action. We define $p(x, a, y)$ as the probability of moving from state x to state y when taking action a. These probabilities depend on the event and the policy. Types of events are arrivals of a certain type of call and service completions of a certain type of agent from a certain group. In our model it holds that $p(x, a, y) \in [0, 1]$. Randomness is involved because the inter-arrival times and service times of jobs are stochastic. Note

that when assuming non-randomized routing policies, the ordering in which the arrivals and service completions occur determines exactly the evolution of the state over time.

The policy of our model needs to be further formalized. We define a policy by a vector π, consisting of different components, and having exactly one component for each type of event

$$\pi = \left(\pi^a_1, \pi^a_2, \ldots, \pi^a_{M-1}, \pi^a_M, \pi^s_{1,1}, \pi^s_{2,1}, \ldots, \pi^s_{M-1,1}, \pi^s_{M,1}, \pi^s_{1,2}, \ldots\right),$$

with π^a_m specifying the action at an arrival of a type-m job, and π^s_{mg} denoting the action at a service completion in group g of a job of type m. The policy π is a function of the state x, $\pi : x \to a$, with a denoting the action, and the function is denoted by $\pi(x)$. Note that if there is no randomness involved in the action, the action completely determines the state after the transition.

It is well known that optimal policies do not randomize when there are no side constraints. Hence, the number of possible actions in each state x is fixed. At arrival of a job of type m, we let the policy π^a_m specify the agent group to which the job is assigned, then $\pi^a_m(x) \in \{g \in \mathcal{G} : m \in S_g\}$, with 0 denoting that the job is queued. Note that a job of type m can not be queued if $x^q_m = L_m$. Then the job is rejected. At a service completion of type m by an agent in group g, we let π^s_{mg} specify the next activity of the agent, who either becomes idle or takes a job from one of the queues, then $\pi^s_{mg}(x) \in \{m \in \mathcal{M} : x^q_m > 0\}$, with 0 denoting idleness. In special cases, we define $\pi^s_{mg}(x) \in \emptyset$ if $x_{mg} = 0$ for all m. The above description defines completely the set of feasible actions $\mathcal{A}(x)$ for every different type of event. Note that $\mathcal{A}(x)$ is a vector of sets. We call a policy π feasible if $\pi \in \mathcal{A}$, component-wise for each state and event. The set of feasible policies is referred to by Π.

Costs and objective

The purpose of introducing actions is controlling the system such that the long-run weighted-average holding cost of jobs in the queues is minimized. Therefore, we define the holding cost per unit of time in state x as

$$c(x) = \sum_{m \in \mathcal{M}} w_m x^q_m,$$

with x^q_m defined as the length of the queue associated with type m and w_m defined as the cost of having a job of type m during one unit of time in

the system. This was earlier defined as the priority of type-m jobs and now translated to costs. Since each group has a fixed number of agents, it should hold that $\sum_m x_{mg} \leq s_g$, for all $g \in \mathcal{G}$. The state variables x_m^q and x_{mg} are all assembled in the state vector x. The size of the state space is

$$\prod_{m=1}^{M} (L_m + 1) \prod_{g=1}^{G} \begin{pmatrix} s_g + |S_g| \\ s_g \end{pmatrix}. \tag{4.1}$$

Let us consider the process $\{x_n^\pi, n \geq 0\}$, with x_n^π the state after transition n under policy π. The weighted long-run average holding cost in the system until the k-th transition, occurring at time T, under policy π is

$$g^\pi = \lim_{k \to \infty} \frac{\sum_{n=0}^{k} c(x_n^\pi)}{k+1}$$

in discrete time, and

$$g^\pi = \lim_{k \to \infty} \frac{\int_{t=0}^{T} c(x^\pi(t))dt}{T}$$

in continuous time, with $x^\pi(t)$ the state at time t under policy π. One of our most important objectives in this thesis is to find a policy that minimizes the long-run average holding costs of the waiting customers in each queue. Hence, we define an optimal policy π^* by

$$\pi^* = \arg\min_{\pi \in \Pi} g^\pi.$$

The next sections of this chapter are devoted to the relevant techniques that are available in the literature.

4.3 Derivation of routing policies

To obtain optimal routing policies one needs to measure the performance of the system. This can be achieved by calculating the expected future costs of all possible starting states. If future rewards are discounted there exists in our model one solution to these future costs. Whereas, in undiscounted

Markov processes the future costs accumulate to infinity. In the latter case only the differences between the values of states have a meaning. A vector with the relative differences between the values of all states is called a bias vector. In this thesis we consider undiscounted Markov processes.

4.3.1 Poisson equation

In this section we introduce a method to measure the performance of a policy π. The method solves the value function (also called bias function). Next to performance measurement, the method is also useful for optimization purposes.

The bias vector v of an undiscounted continuous-time Markov reward process is usually obtained by solving the Poisson equation

$$ge = c + Q^{\pi}v, \tag{4.2}$$

with g the average cost per time unit, e the unity vector, c containing the immediate costs $c(x)$ of all states, and Q^{π} a matrix with the infinitesimal transition rates associated with policy π. We assume an ordering and mapping of the states to indices that denote the row and column in Q^{π} for each state. Then in case of our contact center model we have to solve

$$\begin{aligned} g = \quad & c(x) - \left[\sum_{m \in \mathcal{M}} (\lambda_m + \sum_{g \in \mathcal{G}} \mu_{mg} x_{mg}) \right] v(x) \\ & + \sum_{m \in \mathcal{M}} \lambda_m v(x + \mathrm{I}_{r=0} e^q_m + \mathrm{I}_{r \neq 0} e_{mr}) \\ & + \sum_{m \in \mathcal{M}} \sum_{g \in \mathcal{G}} \mu_{mg} x_{mg} v(x_{mg} - e_{mg} + \mathrm{I}_{m' \neq 0}[e_{m'g} - e^q_{m'}]), \end{aligned}$$

with $r \equiv \pi^a_m(x)$ and $m' \equiv \pi^s_{mg}(x)$.

Equivalently, after uniformization the bias vector v can be obtained by solving the equation

$$v + \tilde{g}e = \tilde{c} + P^{\pi}v, \tag{4.3}$$

with P^{π} the transition probabilities under policy π, and $\tilde{g} = g/\alpha$ and $\tilde{c} = c/\alpha$. The values g and c are redefined by $\tilde{g}$ and $\tilde{c}$ because the time-scale is changed. This equation is derived from the Poisson equation by dividing all components by α.

We remark that an alternative method for performance measurement is to solve the balance equations, see Section 4.3.2. However, the method is not useful for optimization purposes.

4.3.2 Solution techniques

This section treats standard technique to solve the Poisson equation in a numerical way. We discuss a linear programming approach and the so-called power series algorithm.

Another relevant technique is value iteration. This technique is not explained in this section because it is especially useful for the optimization of policies and, hence, discussed in Section 4.3.3.

Linear programming

A linear mathematical programming model to solve Equation (4.3) is given by

$$\begin{aligned} &\min \tilde{g} \\ &\text{subject to} \\ &v(x) + \tilde{g} \geq \tilde{c}(x) + \sum_{y \in \mathcal{X}} p(x, \pi(x), y) v(y), \quad \forall x \in \mathcal{X}, \\ &\tilde{g} \in \mathbb{R}_+, v(0) = 0, \text{ and } v(x) \geq 0, \qquad \forall x \in \mathcal{X}, \end{aligned} \tag{4.4}$$

with $p(x, \pi(x), y)$ the element in the row and column of P^π associated with x and y, respectively. The corresponding dual problem is

$$\begin{aligned} &\max \sum_{x \in \mathcal{X}} \tilde{c}(x) d(x) \\ &\text{subject to} \\ &\sum_{x \in \mathcal{X}} d(x) \leq 1, \\ &\sum_{x \in \mathcal{X}} d(x) p(x, \pi(x), y) = d(y), \quad \forall y \in \mathcal{X}, \\ &d(x) \geq 0, \qquad \forall x \in \mathcal{X}. \end{aligned}$$

For additional information we refer to Kallenberg (2002).

Power series algorithm

The power series algorithm (PSA) was introduced by Hooghiemstra, Keane, and Van de Ree (1988), in order to compute the equilibrium probabilities in a numerically efficient way, i.e., the solution of $0 = \pi Q$ such that $\pi e = 1$. In this section we apply the power series algorithm in order to find a solution of the Poisson equation (4.2). A description is given next.

The algorithm requires to replace the variables $v(x)$ and g by power series of a system parameter, denoted as λ. This parameter must be chosen in such a way that the Markov chain has transition probabilities of the form

$$q_{ij} = \lambda^{(L(j)-L(i))^+} q'_{ij},$$

and q'_{ij} independent of λ, with L a level function, $L : \mathcal{X} \to \mathbb{N}$. By level k we denote all states $x \in \mathcal{X}$ such that $L(x) = k$. We assume that the level function has the following properties:

1. Level 0 consists of one state;

2. The states within each level can be ordered such that there are no transitions to higher ordered states within that level;

3. Transitions to lower level states are possible in each state unless $L(x) = 0$.

In our queueing system the conditions are satisfied by defining L as the number of jobs in the system and by choosing λ equal to the arrival rate, explaining our choice for the symbol λ. Hence, we define the power series of $v(x)$ and g by

$$v(x, \lambda) = \sum_{k=0}^{\infty} v^{(k)}(x)\lambda^k, \tag{4.5}$$

and

$$g(\lambda) = \sum_{k=0}^{\infty} g^{(k)}\lambda^k. \tag{4.6}$$

The coefficients $v^{(k)}(x)$ and $g^{(k)}$ are solved by replacing in Equation (4.2) v and g by power series of λ as in (4.5) and (4.6), and replacing λ_m by $\lambda_m\lambda$,

resulting in

$$\begin{aligned}\sum_{k=0} g^{(k)}\lambda^k = \quad & c(x) - \sum_{k=0}^{\infty}\left[\sum_{m\in\mathcal{M}}(\lambda_m\lambda + \sum_{g\in\mathcal{G}}\mu_{mg}x_{mg})\right] v^{(k)}(x)\lambda^k \\ & + \sum_{m\in\mathcal{M}}\sum_{k=0}^{\infty}\lambda_m\lambda v^{(k)}(x + \mathrm{I}_{r=0}e_m^q + \mathrm{I}_{r\neq 0}e_{mr})\lambda^k \\ & + \sum_{m\in\mathcal{M}}\sum_{g\in\mathcal{G}}\sum_{k=0}^{\infty}\mu_{mg}x_{mg}v^{(k)}(x_{mg} - e_{mg} + \mathrm{I}_{m'\neq 0}[e_{m'g} - e_{m'}^q])\lambda^k.\end{aligned}$$

The order-k terms are

$$\begin{aligned}g^{(k)} = \quad & c(x)\mathrm{I}_{k=0} - \sum_{m\in\mathcal{M}}\lambda_m v^{(k-1)}(x) - \sum_{m\in\mathcal{M}}\sum_{g\in\mathcal{G}}\mu_{mg}x_{mg}v^{(k)}(x) \\ & + \sum_{m\in\mathcal{M}}\lambda_m v^{(k-1)}(x + \mathrm{I}_{r=0}e_m^q + \mathrm{I}_{r\neq 0}e_{mr}) \\ & + \sum_{m\mathcal{M}}\sum_{g\in\mathcal{G}}\mu_{mg}x_{mg}v^{(k)}(x_{mg} - e_{mg} + \mathrm{I}_{m'\neq 0}[e_{m'g} - e_{m'}^q]).\end{aligned} \tag{4.7}$$

The power series algorithm starts with the initialization: $g^{(0)} = 0$ and $v^{(0)} = 0$. During the algorithm, all coefficients of order 1,2,... are determined recursively. For solving order k, it solves $g^{(k)}$ from (4.7) in state $x = 0$, yielding

$$\begin{aligned}g^{(k)} = \quad & -\textstyle\sum_{m\in\mathcal{M}}\lambda_m v^{(k-1)}(x) \\ & +\textstyle\sum_{m\in\mathcal{M}}\lambda_m v^{(k-1)}(x + \mathrm{I}_{r=0}e_m^q + \mathrm{I}_{r\neq 0}e_{mr}).\end{aligned}$$

Next, it iterates on all x within each level and solves $v^{(k)}(x)$ by rewriting (4.7) as

$$\begin{aligned}v^{(k)}(x) = & -g^{(k)} + c(x)\mathrm{I}_{k=0} - \sum_{m\in\mathcal{M}}\lambda_m v^{(k-1)}(x) \\ & + \sum_{m\in\mathcal{M}}\lambda_m v^{(k-1)}(x + \mathrm{I}_{r=0}e_m^q + \mathrm{I}_{r\neq 0}e_{mr}) \\ & + \sum_{m\in\mathcal{M}}\sum_{g\in\mathcal{G}}\mu_{mg}x_{mg}v^{(k)}(x_{mg} - e_{mg} + \mathrm{I}_{m'\neq 0}[e_{m'g} - e_{m'}^q])/\sum_{m\in\mathcal{M}}\sum_{g\in\mathcal{G}}\mu_{mg}x_{mg}.\end{aligned}$$

We note that the secondly mentioned property states that an ordering of the states is required. For our model any ordering of the states within a level suffices.

Once the coefficients are calculated, the values and average cost can be obtained by taking $\lambda = 1$ in (4.5) and (4.6).

Matrix multiplication

A related method that is relevant to measure the performance of a routing policy π, but that does not solve the Poisson equation, is matrix multiplication, see for example Tijms (1986a). The method is useful to calculate the average cost, by using

$$g^{\pi} = \sum_{x \in \mathcal{X}} c(x)p(x),$$

with $p(x)$ denoting the stationary probability that the process is in state x. To determine the vector p we need to solve the so-called balance equations

$$p = pP^{\pi}.$$

This can be achieved iteratively by

$$p_n = p_{n-1}P^{\pi} = p_0(P^{\pi})^n,$$

for some initial vector p_0 such that $\sum_{x \in \mathcal{X}} p(x) = 1$. Although P^{π} is a sparse matrix in the application to call centers, the matrix $(P^{\pi})^n$ becomes more dense when n increases, such that it involves many computations. The series $(p_n : n \geq 0)$ converges to the vector of stationary probabilities p when n goes to infinity.

4.3.3 Optimization techniques

This section discusses several techniques to obtain optimal routing policies in a numerical way. This is achieved by means of successive policy improvements. A policy is defined to be optimal if it gives a solution to the Bellman equation

$$v + \tilde{g}e = \min_{\pi \in \Pi} \{\tilde{c} + P^{\pi}v\}. \tag{4.8}$$

The techniques that we discuss are called dynamic programming, policy iteration, and linear programming.

Dynamic programming algorithm

The best-known method to solve Equation (4.8) is dynamic programming, also known as backward recursion and value iteration, which is an iterative method. It starts with the initialization $v_0(x) = 0$, for all $x \in \mathcal{X}$. Next,

vectors $v_1, v_2, \ldots$ are computed iteratively until some stopping criterion is satisfied. In the n-th iteration, the vector v_n is computed according to

$$v_n = \min_{\pi_n \in \Pi} \{\tilde{c} + P^{\pi_n} v_{n-1}\}.$$

This algorithm produces a sequence of policies $\pi_1, \pi_2, \ldots$ that is optimal in each iteration. After each iteration, the absolute differences in the values between two consecutive iterations, $v_n(x)$ and $v_{n-1}(x)$, are calculated for all states x. The algorithm stops if the gap between the minimum and maximum difference over all states is smaller than a small number. Afterwards, the average cost $\tilde{g}$ can be determined by $v_n(x) - v_{n-1}(x)$, for some state $x \in \mathcal{X}$.

Policy iteration

Policy iteration consists of two steps that are executed consecutively: an evaluation step and an improvement step. The algorithm starts with an initial policy π_0, and $n = 0$. The two steps are as follows:

1. Obtain v_n by solving (4.2) or (4.3) and using policy π_n.

2. Determine the improved policy

$$\pi_{n+1} = \arg\min_{\pi \in \Pi} \{\tilde{c} + P^{\pi_n} v_n\}, \tag{4.9}$$

   and set $n = n + 1$. Stop if π_{n+1} and π_n are identical, otherwise, go to step 1.

The policies $\pi_0, \pi_1, \ldots$ are a sequence of improving policies and we conjecture that this sequence converges to optimality. According to the theory of Markov decision processes an optimal policy is found if π_n and π_{n-1} are identical, i.e., both policies dictate the same action for every state and every event. The improved policy that results from one policy iteration is usually called a one-step improved policy, and the method itself is referred to as one-step policy improvement.

Linear programming

An optimal policy is determined by the dual solution of

$$\min \tilde{g}$$

subject to

$$\begin{aligned} &\tilde{g} + v(x) \geq \tilde{c}(x) + \sum_{y \in \mathcal{X}} p(x,a,y) v(y), \quad \forall x \in \mathcal{X}, \forall a \in \mathcal{A}(x), \\ &g \geq 0, \qquad v(x) \geq 0, \qquad \forall x \in \mathcal{X}. \end{aligned}$$

The decision variables are $\tilde{g}$ and all $v(x)$. The corresponding dual problem is

$$\max \sum_{x \in \mathcal{X}, a \in \mathcal{A}(x)} \tilde{c}(x) d(x,a)$$

subject to

$$\begin{aligned} &\sum_{x \in \mathcal{X}, a \in \mathcal{A}(x)} d(x,a) \leq 1, \\ &\sum_{x \in \mathcal{X}, a \in \mathcal{A}(x)} d(x,a) p(x,a,y) = \sum_{a \in \mathcal{A}(y)} d(y,a), \quad \forall y \in \mathcal{X}, \\ &d(x,a) \geq 0, \qquad \forall x \in \mathcal{X}, \forall a \in \mathcal{A}(x). \end{aligned}$$

We remark that solving the primal problem is sufficient to obtain an optimal policy, because simplex algorithms also provide the solution of the dual problem.

4.4 Numerical examples

We consider a call center with two skills, which is depicted in Figure 4.1. The arrival rates of the two job types are $\lambda_1 = \lambda_2 = 2.5$ and the weights are $w_1 = 0.5$ and $w_2 = 2$, respectively. There are two specialists available for each job type and these four specialists complete the service of jobs with rate 1, while four generalists serve with rates $\mu_{13} = 0.25$ and $\mu_{23} = 1.75$ jobs of type 1 and 2, respectively.

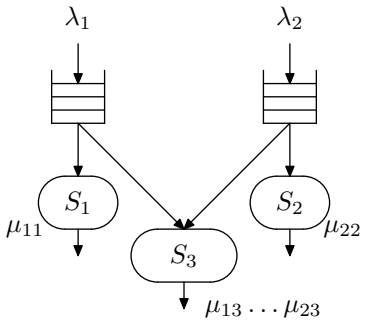


Figure 4.1: Model of a call center with two skills

The routing problem of this call center instance is well tractable, since there are only two skills and eight agents. For $L_1 = L_2 = 50$ the total number of states is $51^2 \times 3^2 \times 15 = 351135$, calculated according to (4.1). Notice that this number can be reduced by assuming that specialists always work if jobs are available. This allows us to count the number of jobs in the queue and the number of busy specialists for each type by only one decision variable, reducing the number of states to $53^2 \times 15 = 42135$. We calculated the optimal policy π^* by applying both value iteration and the power series algorithm. The average cost of this policy g^{π^*} turned out to be 3.6 per time unit.

The optimal policy is complicated to describe. We found that the following properties hold in general. Specialists get assigned as much work as possible. Generalists in group 3 prefer jobs of the type $m \in \{1, 2\}$ for which the value $w_m \mu_{3m}$ is the highest. The jobs of the other type are queued if there is a substantial number of preferred jobs in the system and the queue length of the non-preferred job type is moderate.

Relatively larger call centers with two skills and also call centers with more skills and a moderate number of agents become intractable. For example, a call center with three skills and ten specialists of each type, ten generalists, and $L_1 = L_2 = L_3 = 50$ yields a state space of 64916566. In this case, the state space is reduced by expressing the sum of the queue length and the number of busy specialists of each job type in one variable.

4.5 Analytical methods

Analytical methods can be helpful for obtaining optimal routing policies for the assignment of calls to agents. For the general model of the multi-skill call center from Chapter 2, several structural results are available in the literature. For example, we are familiar with Chevalier, Shumsky, and Tabordon (2004), in which structural results are obtained for a system with specialists and generalists. However, queues are omitted in their model. For other structural results of special cases we refer to Section 3.6.

A promising technique to obtain structural results is stochastic coupling, also called sample path analysis, and discussed next.

Stochastic coupling

For an introduction to stochastic coupling we refer to Serfozo (1999) and El-Taha and Stidham (1998). However, the basic idea is not very difficult to understand and readers with a background in stochastics will probably encounter no difficulties while reading this section.

In the remainder of this section we consider a multi-skill call center, conform Chapter 2, with exponential distributions, equal service rates among all job types, equal weights, non-preemptive service disciplines, and no queues ($L_m = 0$). Agents with the same skills are grouped together. We allow jobs to be rejected, which can not be avoided if there is no agent with the required skill available. Rejected jobs abandon the system.

The service level is defined as the percentage of jobs that are admitted to the system, and we consider optimal policies.

Theorem 4.1. *The service level decreases when an agent is moved to a less-skilled group.*

Proof. Consider the situation in which exactly one agent becomes less skilled. We define the optimal routing strategy of the original and the new setting as π^* and $\bar{\pi}^*$, respectively. It is obvious that policy $\bar{\pi}^*$ can be applied to the original setting, with the agent using the skills from the original setting. Thus, the performance of π^* is at least as good as $\bar{\pi}^*$. □

We remark that Theorem 4.1 also holds with overflow routing when specialists have a higher priority than generalists in the assignment of calls to agents.

Theorem 4.2. *The service level is optimized if each agent serves all types of jobs that he or she is capable of.*

Proof. Follows directly from Theorem 4.1. □

Theorem 4.3. *According to the optimal policy, jobs are always admitted to increase the service level.*

Proof. We prove the theorem by coupling arguments. We have to show that when being in a state, say x, and a job is admitted to group i, that the number of future rejections of $x + e_i$ will not exceed the number of future rejections of x by more than 1, meaning that admission is better than rejection. This is easy to show by comparing two coupled processes that start in both states. If the optimal policy assigns a job to group i we can reject the job in the system with the additional busy server, making the total number of rejections in both systems equal. If the job in group i finishes before the rejection occurs, both systems become the same too and a rejection is no longer necessary. □

Theorem 4.4. *If two agents are available and the skill set of one agent is a subset of the other, the optimal policy assigns the job to the agent with the fewest skills.*

Proof. Consider two identical systems that are coupled stochastically. Policy π^* denotes the optimal policy in the system that assigns jobs to an agent with additional skills compared to the skills of another available agent or to an agent with the same skills. This policy can also be applied to the system that assigns jobs to the agents with the fewest skills. Of course, if jobs are not admitted to the first system, they are also rejected in the second system. □

4.6 Concluding remarks

In this chapter we aimed to illustrate the limitations of standard techniques from the literature. In particular, it explained the limitations of the standard numerical methods from Markov decision theory by means of two examples. In addition, we showed that analytical methods have limitations by pointing out that there are not many results available in the literature with regard

to multi-skill and multi-group call centers (while many researchers are interested in this subject). The limitations are caused by the curse of dimensionality, referring to the exponential grow of the states of a system as the number of skills and agent groups grow. Hence, we consider approximation methods in the remainder of this thesis.

Chapter 5

Routing by Approximate Dynamic Programming

In this chapter we apply approximate dynamic programming to our call center model from Chapter 2. Our objective is to obtain good routing policies that determine the assignment of jobs to the multi-skilled agents. The jobs either newly arrive or are taken from the queues. Assignments take place at job arrivals and service completions by agents.

5.1 Introduction

A multi-skill call center is considered, having specialists and generalists, and skill- and group-dependent service rates. We define specialists as agents with exactly one skill and generalists as agents with all skills. Group-dependent service rates are realistic because specialists often work faster than generalists. In addition, different priorities are assigned to the call types. A priority is modeled as a cost for holding a call in the queue during one unit of time. The holding costs for high-priority calls are considered higher than for low-priority calls. By minimizing these costs, the queue length of jobs with a high priority will decrease and waiting times will get shorter, which increases the performance. We are interested in the associated routing policies for calls. These policies dictate the assignment of calls to agents.

Good routing policies help call centers to reduce the number of calls in the system, which is a way to improve the service level. Consider a multi-skill call center, with agents having different skill sets. It is common that at an arrival of a call different agents with different skill sets are available

that can all handle the call. The decision about the agent that is chosen influences the long-run average number of jobs in the system.

Obtaining optimal routing policies for multi-skill call centers is a difficult problem to solve analytically. Hardly any structural results exist. Some are given in Chevalier, Shumsky, and Tabordon (2004) (in Appendix A). However, they consider a model without queues and without priorities. (They do consider group- and skill-dependent service rates.) Their results are difficult to extend to our model.

We model the system as a Markov process and, hence, the determination of routing policies is formulated as a Markov decision problem. Dynamic programming (DP), see Chapter 4, offers techniques to solve Markov decision problems, both analytically and numerically. The techniques are potentially capable to numerically determine dynamic routing policies of multi-server queueing systems. However, a shortcoming of DP is the fact that high-dimensional systems, such as most call centers, are not tractable because the state space is too large. This is called the curse of dimensionality, which is introduced in the context of Markov processes in Bellman (1961). By using standard techniques we were not able to solve examples with more than ten agents and two skills. The methods that we considered are value iteration and policy iteration, and policy iteration in conjunction with the power series algorithm and matrix multiplication. We refer to Section 4.3 for literature on these methods, and to Koole and Pot (2006c) for an example of the power series algorithm to obtain policies for high-dimensional queueing systems.

This chapter addresses call centers that we are not able to analyze by using the standard techniques from Chapter 4. Hence, we explore the potential of approximate dynamic programming to obtain results for these call centers. We will show that approximate dynamic programming is an efficient and effective way to obtain dynamic routing policies. The advantage of approximate dynamic programming is that it suffers less from the curse of dimensionality, mainly because it allows us to traverse only a subset of all states.

In the literature, it has also been shown several times that approximate dynamic programming can reduce the numerical complexity without much loss of performance and accuracy. It reduces both the computation time and the memory usage. The last one becomes almost zero. The game Tetris is an example of a successful implementation and treated in Farias and Van Roy (2006). However, in some cases approximate dynamic programming was not

successfully applied, some are mentioned in De Farias and Van Roy (2004). There are several reasons, for example the computation times get too long or the value function has a structure that is difficult to approximate. This will be explained at a later point in this chapter, after the different techniques have been discussed. More references to the literature about approximate dynamic programming are also given at a later point in this chapter.

The method that we use for the optimization is policy improvement in conjunction with an approximation method, being discussed in Section 5.3. However, we will give a brief description and introduction. Policy improvement starts with calculating the value function, well known from DP, under an initial policy. It is obtained by solving the so-called Poisson equation. The solution of this equation gives next to the value function also the average cost of the system, which will serve as a performance measure of the system. Next, policy improvement is executed and this yields an improved policy. Afterwards, the system performance is measured under the policy resulting from the improvement step. Several methods to measure performance will be discussed.

In the context of approximate dynamic programming, the solution of the Poisson equation is approximated by replacing the value function by a simple structure and by fitting the structure as close as possible to the real value function. The approximation of the value function is used in the one-step policy improvement. The techniques that we consider for the approximation are presented in Section 5.4.

The content of this chapter is structured as follows. Section 5.2 describes the model and discusses the control parameters and the dynamics. In Section 5.3 the call center is modeled as a Markov Decision Process (MDP) and policy improvement is discussed. Section 5.4 treats an introduction to approximate dynamic programming. Next, we describe:

- a numerical analysis in order to find a structure for the value function such that policy improvement is most effective (Section 5.5),
- the estimation of the coefficients of the approximation function by minimizing the so-called Bellman error and by using the linear programming formulation of the Poisson equation (Section 5.6),
- and numerical results of call centers with two and three skills (Section 5.7).

Concluding remarks and plans for future research are given in Section 5.8.

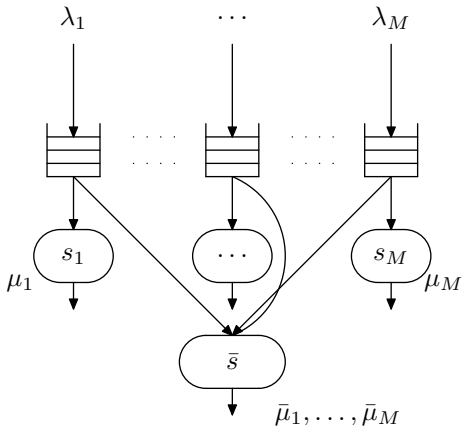


Figure 5.1: Model of a call center with M skills

Although our model provides flexibility only by means of one group of generalists, high service levels can still be obtained. Chevalier, Shumsky, and Tabordon (2004) show that just a little bit of flexibility can improve the service level a lot. Their conclusion is that optimal staffing requires that twenty percent of all agents are cross-trained, and that the others are specialist. This shows that, despite the fact that our model includes flexibility by having only one group of generalists, high performance can still be obtained. Concerning routing, she considers only a specific type of routing policies; calls are first assigned to specialists and, if all specialists are busy, calls are assigned to generalists. They prove that this policy is optimal in blocking systems with specialists working faster than generalists and without call priorities. However, the queues and call priorities from our model make routing much more complicated. This chapter proposes a method to obtain routing policies for these cases.

5.2 Model

The model, with M skills, is depicted in Figure 5.1. The jobs of each type arrive according to independent Poisson processes. There is one group of specialists for each type and one group of generalists, who all have M skills. Jobs of each skill type are served according to a first-come-first-serve service discipline. Each job is served exactly once, either by a specialist or

a generalist. This is determined by the routing policy and is discussed in Section 5.3.

The service times are exponentially distributed and the parameter of the group with specialists of type m is μ_m, with $m \in \{1, 2, \ldots, M\} =: \mathcal{M}$. Generalists work successively on calls of different types, with parameter $\bar{\mu}_m$ in the case of a type-m call. The service rates are not only skill-dependent but also group-dependent, because in reality specialists often work faster than generalists. However, the model also allows generalists to work faster than specialists.

The number of specialists in the group with skill m is denoted by s_m and $\bar{s}$ denotes the total number of generalists. The state space is defined in such a way that it is possible that calls are directed to generalists while keeping specialists idle, and vice versa. It consists of $2M$ components

$$x := (x_1, x_2, \ldots, x_M, \bar{x}_1, \bar{x}_2, \ldots, \bar{x}_M),$$

and x_m is the total number of jobs in the queue and the number of jobs served by specialists of group m. The variable $\bar{x}_m$ is the number of generalists that work on type-m jobs. Because there are $\bar{s}$ generalists we have $\bar{x}_1 + \bar{x}_2 + \ldots + \bar{x}_M \leq \bar{s}$. According to the definition of x_m it is not possible to queue jobs while keeping specialists idle. We denote the state space by $\mathcal{X}$, which contains all feasible states.

There are linear holding costs w_m for each type. The objective of our analysis is to find a non-preemptive routing policy that minimizes the long-run average holding costs of the waiting customers in each queue. The control policy is described in the next section. The minimization of holding costs is a difficult problem, because generalists can work on different job types. If there is no job available with the highest priority at a service completion by a generalist, there is a trade-off between serving a lower-priority job and waiting for a high-priority job.

Our model is an extension of the one studied in Örmeci (2004), which treats the case of $M = 2$ without waiting rooms. Instead of holding costs, different rewards for each type of call are considered. She shows that under certain conditions the optimal policy of the assignment of calls to generalists can be characterized as a monotonic threshold policy. In that paper no answer is given to the question if the optimal policy is work-conserving, because a loss model is considered. Although the paper is relevant, the structural results appeared not to be useful for our problem, because adding queues and considering additional job types complicate the problem a lot.

Moreover, there is a big difference between admission and routing; in our model, all jobs are admitted to the system, while the routing policy affects the order in which jobs are served.

Policy

The performance of our call centers is controlled by a routing policy, denoted by π. We refer to the space of all feasible policies by Π and to the set of feasible policies in state x by $\Pi(x)$. A routing policy dictates the assignment of calls to agents, and is state-dependent. In a particular state, the policy denotes an action, $\pi : x \rightarrow a$, and $p(x, a, y)$ is the probability of going from state x to state y according to action a. Each action describes the transition in case of an arrival, a service completion by a specialist, and a service completion by a generalist, for each type of call. Hence, an action is a vector that is composed of three types of elements, $\pi(x) = (\pi_1^a, \ldots, \pi_M^a, \pi_1^s, \ldots, \pi_M^s, \bar{\pi}_1^s, \ldots, \bar{\pi}_M^s)(x)$. The different elements will be discussed next.

Arrival: The job-assignment policy at an arrival of type m in state x is defined as

$$\pi_m^a(x) := \begin{cases} 0 : \text{generalist} \\ 1 : \text{specialist or queued} \end{cases}.$$

The value 0 denotes that the job of type m is assigned to a generalist, resulting in a transition to state $x + \bar{e}_m$. If the action is 1, two possibilities exist, depending on the state: the job is assigned to a specialist if one is available, or queued. Sometimes only one action is possible, for example when $s_1 = 0$ or $\bar{x}_1 + \bar{x}_2 + \ldots + \bar{x}_M = \bar{s}$. To prevent that the system reaches infeasible states, we always take

$$\pi_m^a(x) = \begin{cases} 1 : \sum_{m \in \mathcal{M}} \bar{x}_m = \bar{s} \\ 0 : s_m = 0, \sum_{m \in \mathcal{M}} \bar{x}_m < \bar{s} \end{cases}.$$

Specialist completes: At a service completion by a specialist having skill m in state x there are two possibilities

$$\pi_m^s(x) := \begin{cases} 0 : \text{idle} \\ 1 : \text{type } m \end{cases}.$$

The value 0 indicates idling and in the case of value 1 the specialist receives another job of type m. Note that if a specialist of type m finishes a job and the queue is not empty, the decision about the assignment of the first call in the queue is the same as when this call would arrive instantaneously.

Generalist completes: At a service completion by a generalist in state x the generalist can choose several actions

$$\bar{\pi}_m^s(x) := \begin{cases} 0 : \text{idle} \\ j : \text{type } j \\ \text{-1: random selection from set } \mathcal{M} \end{cases} .$$

This choice depends on the job type of the finished call. We note that an optimal policy does not necessarily randomize. Randomization is supported because we prefer that the initial policy is randomized. In the first case (0) the generalist becomes idle and the new state is $x - \bar{e}_m$, meaning that a generalist that serves a type-m call finishes. In the second case (j) a call from queue j is assigned, with a transition to $x - e_j - \bar{e}_m + \bar{e}_j$, meaning that the queue length of type-j jobs becomes one job shorter, a generalist that serves a type-m job finishes and a generalist starts to serve a job of type j. Finally, the value -1 denotes a random selection from the job types $1, 2, \ldots, M$ with non-empty queues, with equal probability. A special case is

$$\bar{\pi}_m^s(x) = \begin{cases} 0 : x_m = 0, \forall i \\ 0 : \bar{s} = 0 \end{cases} .$$

5.3 Optimization

The dynamics of the system can be described by using dynamic programming. An extensive tutorial and overview of Markov Decision Processes is given in Puterman (1994). We refer to this book and to Chapter 4 for general background information on dynamic programming and for algorithms to improve policies. An application to a call center is discussed in Chapter 4. Methods to approximate the value function are the subject of Section 5.4.

The technique that we apply in this chapter is called policy iteration. It consists of an evaluation step, which yields the value function, and an improvement step. These two steps can be executed consecutively.

Policy evaluation

The value function associated with policy $\pi \in \Pi$ can be expressed as the solution of the Poisson equation

$$v(x) + \tilde{g} = \tilde{c}(x) + \sum_{y \in \mathcal{X}} p(x, \pi(x), y) v(y), \qquad \forall x \in \mathcal{X}, \tag{5.1}$$

with $p(x, \pi(x), y)$ the transition probability (after uniformization, see Section 4.2) of going from state x to state y according to action $\pi(x)$, $\tilde{c}(x)$ is the immediate cost of the transition, $\tilde{g}$ the average cost, and $v(x)$ the value of state x. In more detail, we have (by moving all terms of $v(x)$ to the left-hand side)

$$\begin{aligned}
&\sum_{m \in \mathcal{M}} (\lambda_m + \mu_m \min\{x_m, s_m\} + \bar{\mu}_m \bar{x}_m) v(x) + \tilde{g} = \tilde{c}(x) + \\
&\sum_{m \in \mathcal{M}} (\lambda_m v(x + e_m) \pi^a_m(x) + \lambda_m v(x + \bar{e}_m)(1 - \pi^a_m(x))) + \\
&\sum_{m \in \mathcal{M}} (\mu_m \min\{x_m, s_m\} v((x - e_m)^+) + \\
&\bar{\mu}_m x_m [v((x - \bar{e}_m)^+) \mathrm{I}\{\bar{\pi}^s_m(x) = 0\} + \\
&\sum_{j \in \mathcal{M}} v((x - e_j - \bar{e}_m + \bar{e}_j)^+) \mathrm{I}\{\bar{\pi}^s_m(x) = -1\}/M \\
&\sum_{j \in \mathcal{M}} v((x - e_j - \bar{e}_m + \bar{e}_j)^+) \mathrm{I}\{\bar{\pi}^s_m(x) = j\}]),
\end{aligned}$$

with the cost function $\tilde{c}(x)$ defined as

$$\tilde{c}(x) := \sum_{m \in \mathcal{M}} \tilde{w}_m (x_m - s_m)^+,$$

i.e., the holding costs times the queue lengths. The parameters $\tilde{w}_m$ is equal to w_m divided by the uniformization parameter, see Section 4.2 for an explanation.

Jobs of type m arrive with rate λ_m and are assigned to a specialist or a generalist, depending on action π^a_m. The group with specialists of type m finish jobs with rate $\min\{x_m, s_m\}\mu_m$ when there are $\min\{x_m, s_m\}$ agents busy and $(s_m - x_m)^+$ agents idle. Generalists can work on both types of jobs. When they finish a job of type m, a decision is made about the next type of job, which is specified by action $\bar{\pi}^s_m$.

Policy improvement

Standard policy improvement methods from the literature, discussed in Chapter 4, require that the system is evaluated under an initial policy π_0 before the policy improvement takes place. The initial policy that we consider in our numerical examples is defined as follows. A call is first routed to a specialist. If all specialists with the required skill are busy, then the call is routed to a generalist. If all generalists are busy too, the call is queued. When a generalist completes a call, the next call is chosen randomly from one of the non-empty queues.

Given the initial value function, denoted as v_0, the actions of the one-step improved policy are defined by:

$$\pi_1(x) := \arg \min_{a \equiv \pi(x) \in \Pi(x)} \{\tilde{c}(x) + \sum_{y \in \mathcal{X}} p(x,a,y) v_0(y)\}, \qquad \forall x \in \mathcal{X},$$

such that $\pi \in \Pi$. The right-hand side contains the relative values of each state. These values correspond to the process that starts in state x, with the first action depending on the initial values $v_0(x)$, and, next, evolving according to the initial policy π_0.

Arrivals: Concerning arrivals in state $\{x : s_m > x_m, \bar{x}_1 + \bar{x}_2 + \ldots + \bar{x}_M < \bar{s}\}$, the determination of the optimal action $\pi^a_m(x)$ is not trivial, since an assignment to either a specialist or a generalist is possible. If $\mu_m > \bar{\mu}_m$ and the holding costs are equal, we conjecture that it is optimal to route the arriving call first to a specialist, as shown in Chevalier, Shumsky, and Tabordon (2004) for the blocking model. Consider a system in which a job is queued or is assigned to a generalist while a specialist is idle. Instead, by assigning the same job to a specialist, the expected service time decreases and a generalist becomes also available for other job types, such that the amount of remaining work is minimized.

We will also mention a few situations in which the routing problem is difficult. In the first place, in states that satisfy $x_m < s_m$, $\bar{x}_1 + \bar{x}_2 + \ldots + \bar{x}_M < \bar{s}$ and $\mu_m < \bar{\mu}_m$ for some m, specialists and generalists are available and generalists work faster. In the second place, in states that satisfy $\bar{x}_1 + \bar{x}_2 + \ldots + \bar{x}_M < \bar{s}$, $\mu_{m'} > \bar{\mu}_{m'}$ for multiple m', $x_m \geq s_m$, and different holding costs or service rates per type. Then it is possible that the optimal policy is not work-conserving. The optimal action, with respect to the value function

v, is determined according to

$$\pi_m^a(x) = \begin{cases} 0 : v(x + \bar{e}_m) > v(x + \bar{e}_m) \\ 1 : v(x + e_m) \leq v(x + e_m) \end{cases} .$$

In these inequalities, the differences between future costs of different states are compared, by using the value function. Each value corresponds to the state that results from an action. The action that gives the lowest future costs is chosen. The meaning of 0 and 1 is given is already given in this section. If no specialist or generalist is available the only possible action is to queue the job.

Service completions: Concerning the actions at a service completion by a generalist, the relevant states are $\{x : \bar{x}_m > 0\}$. We introduce

$$v(x, m, j) \equiv v(x - \bar{e}_m + (\bar{e}_j - e_j)), \qquad \forall x_m, x_j > 0,$$

the value of the state after a generalist finishes a type-m job and starts serving a type-j job. The optimal action is determined according to

$$\pi_m^s(x) = \begin{cases} 0 : & v(x, m, 0) < v(x, m, j) & \forall j : x_j > s_j \\ j : & v(x, m, j) \leq v(x, m, k) & \forall k \in 0 \cup \mathcal{M} \end{cases} .$$

The optimal action is determined by the value of the next state of the system, which depends on the action. For example, the server becomes idle if action 0 moves the system to a state with the lowest value, compared to all the other actions.

Heuristics

To further evaluate the difficulty of obtaining good routing policies and also to increase our intuitive understanding, we performed numerical experiments of call centers with two and three different skills. We analyzed if it is possible to obtain nearly optimal performance by using simple heuristic routing rules. From these numerical experiments we conclude that, if specialists are faster than generalists, nearly optimal policies can be obtained by using the following rules:

- Keep specialists busy as much as possible.
- Let a generalist only pick a job from a queue if the queue length exceeds a threshold.

An alternative for the second rule, which we also tested, is reserving generalists for the most important job types, for example by using a threshold on the number of busy generalists, such that jobs with low importance are only served if a sufficient number of agents is available. Note that according to these rules generalists can sometimes be idle while there is work in at least one queue. We compared these rules to the optimal policies obtained by value iteration, explained in Section 4.3.3. Several relatively small call center instances were considered, having two and three skills. These experiments showed that it is possible to obtain policies that are about 15% from optimality.

As we mentioned before, the rules only hold if specialists are faster than generalists. Although it is not very realistic that generalists work faster than specialists, this case is considered (for completeness) in our numerical examples of Section 5.7. We remark that the policies become more complicated in that case. However, we observed that if the difference in service times (per job type) between specialists and generalists is moderate, giving priority to specialists above generalists still yields good policies, also about 15% from optimality. Still, the numerical analysis from Section 5.7 shows that even better policies can be obtained by approximate dynamic programming.

Performance evaluation

Dynamic routing policies can be evaluated by numerical methods. This is relevant to the evaluation of the initial and improved policies. In our analysis the average cost is the main performance measure and analyzed by means of the power series algorithm. The power series algorithm is introduced in Hooghiemstra, Keane, and Van de Ree (1988), and see Blanc (1987) for an overview. For MDPs the method is useful to calculate value functions and stationary probabilities. The power series algorithm solves these values by solving a set of equations, in particular the balance equations or the dynamic programming equations, see Chapter 4. To accomplish this, the unknown values are considered as a polynomial function of a system parameter. Then a power series expansion is applied, such that the unknown coefficients can be calculated recursively. Further details and the application to call centers are discussed in Section 4.3.2. The examples that are not tractable by the power series algorithm in our experiments due to the curse of dimensionality, are evaluated by means of simulation.

5.4 Approximate dynamic programming

Bertsekas and Tsitsiklis (1996) is essentially concerned with approximate dynamic programming methods. These methods are relevant to approximate the solution of Equation (5.1).

Our focus is on two of the methods, namely the Bellman-error method (discussed in Section 5.4.1) and the approximate linear programming method (see Section 5.4.2). Both methods require that a structure for the approximation of v has been chosen in advance, which we denote by $\tilde{v}$. Section 5.5 is devoted to our choice of the structure of $\tilde{v}$. Insight is obtained by fitting different structures of $\tilde{v}$ to the real value function v. Next, based on these experiments we choose to approximate the value function by a polynomial function, and let r denote the vector with unknown coefficients. Each coefficient corresponds to a term of the polynomial function.

The computation time of the fit grows with the size of the state space. In many realistic cases the space is too large, even if the state space is truncated at a moderate level, such that calculations are no longer tractable. Then we are forced to consider a more restricted number of states, denoted by $\tilde{\mathcal{X}}$, and the other states are ignored. This set of states is the so-called set of representative states, see for example De Farias and Van Roy (2004). In Section 5.6 several ways are explored to compose the set of representative states. Finally, the algorithm is presented in the same section.

5.4.1 Bellman-error minimization

In this section we discuss a method for fitting the value function $\tilde{v}$ to the real value function, yielding an approximation of the value function v of Equation (5.1), see also Bertsekas and Tsitsiklis (1996), Chapter 6.10. The method consists of several steps. First, the approximation structure is plugged into the structure of Equation (5.1). Next, the vector r of coefficients is determined, which is achieved in the following way. Let the Bellman error be defined as

$$D(x,r) := \tilde{v}(x,r) - \tilde{c}(x) + \bar{g} - \sum_{y \in \tilde{\mathcal{X}}} p(x,\pi(x),y)\tilde{v}(y,r),$$

being derived from Equation (5.1). The variable $\bar{g}$ denotes an approximation of the average costs. The coefficients are determined by minimizing the

quadratic sum of Bellman errors, which is given by

$$\min_{r\in\mathbb{R}^{2MK}} \sum_{x\in\tilde{\mathcal{X}}} \omega(x)\left(D(x,r)\right)^2,$$

with $\omega(x)$ a weight that we define as

$$\omega(x) := \rho^{xe}, \qquad \rho \in (0,1),$$

with e the unit vector, such that the total number of jobs in the system is counted. The weights are used to make a distinction in the accuracy of the fit between low and high level states. In our numerical analysis, the value of $\bar{g}$ is estimated in advance by means of a simulation run. Hence, this is called a two-phase method, see De Farias and Van Roy (2003).

The remainder of this section deals with the estimation of the unknown coefficients of vector r. The minimization is achieved by applying the Gauss-Newton method, based on a linear approximation of $\tilde{v}(x,r)$ around r. Specifically, we update r to r' according to

$$\begin{aligned} r' = \;& r - \gamma w(x) \sum_{x\in\tilde{\mathcal{X}}} D(x,r)\nabla D(x,r) \\ = \;& r - \gamma w(x) \sum_{x\in\tilde{\mathcal{X}}} D(x,r) \\ & \left(\sum_{y\in\tilde{\mathcal{X}}} p(x,\pi(x),y)\left[\nabla\tilde{v}(y,r) - \nabla\tilde{v}(x,r)\right] \right), \end{aligned}$$

with γ denoting the step size and $\nabla D(x,r)$ the gradient of $D(x,r)$ with respect to r.

The expression $D(x,r)$ is zero when the structure of Equation (5.3) matches the real value function. As an exercise and a verification, we implemented approximate dynamic programming for the $M/M/1$ queue. This yielded accurate results, which is not surprising because it is well known that its value function is indeed a quadratic polynomial function.

5.4.2 Approximate linear programming

In the literature much attention is given to approximate linear programming (ALP), see De Farias and Van Roy (2002). The first paper considers

MDPs with discounted costs, while the second paper treats MDPs with average costs. In this chapter the ALP method is used to approximate the Poisson equation with average costs, see Equation (5.1). It uses the linear programming formulation given by Equation (4.4), which is equivalent to Equation (5.1). The programming formulation has $|\mathcal{X}|$ variables and $|\mathcal{X}|$ constraints, for details and references we refer to Section 4.3.2.

ALP reduces the number of variables and constraints of the linear programming model. Firstly, $v(x)$ is replaced by a function $\tilde{v}(x, r)$, with r a vector of unknown coefficients, which resembles the variables of the new linear program. Secondly, constraints are removed with ALP and only the constraints corresponding to a carefully chosen subset $\tilde{\mathcal{X}}$ remain. The approximate linear program becomes

$$\begin{aligned} &\min_{g \in \mathbb{R}_+, r \in \mathbb{R}^{2MK}} g \\ &\text{subject to} \\ &\tilde{v}(x, r) + g \geq \tilde{c}(x) + \sum_{y \in \tilde{\mathcal{X}}} p(x, \pi(x), y) \tilde{v}(y, r), \quad \forall x \in \tilde{\mathcal{X}}, \\ &\tilde{v}(0, r) = 0, \text{ and } \tilde{v}(x, r) \geq 0, \qquad \forall x \in \tilde{\mathcal{X}}. \end{aligned} \tag{5.2}$$

Concerning the implementation we remark that there are more constraints than variables. If we apply the simplex method, the computation time depends on the size of the basis, which is equal to $|\tilde{\mathcal{X}}|$. Reduction is possible by solving the dual problem of Equation (5.2), see Section 4.3.2. In this representation the size of the basis is equal to $|r| = 2MK$. We use a simplex solver that we have written ourselves to solve the linear programming problems.

5.5 Exploration of approximation structures

In this section, we aim to obtain a polynomial structure $\tilde{v}(x, r)$ for the value function that is accurate enough for one step of policy improvement.

This section is structured as follows. Firstly, the so-called basis function is introduced, together with a discussion of its advantages and disadvantages. Secondly, we discuss our experiments with basis functions. Thirdly, more elaborate structures than basis functions are considered.

A sufficient structure for approximating the value function

We consider value functions according to

$$\tilde{v}(x,r) = \sum_{m\in\mathcal{M}} \sum_{k=1}^{K} (r_m^{(k)} x_m^k + \bar{r}_m^{(k)} \bar{x}_m^k), \tag{5.3}$$

which are called basis functions, see Section 3.1.1 of Bertsekas and Tsitsiklis (1996), with $r_m^{(k)}$ the unknown coefficient of the order-k term that is associated with variable x_m, and $\bar{r}_m^{(k)}$ the unknown coefficient of order k that is associated with variable $\bar{x}_m$. The quality of this structure is analyzed empirically by fitting $\tilde{v}$ to the real value function and measuring the mean-square error

$$\min_{r\in\mathbb{R}^{2MK}} \sum_{x\in\tilde{\mathcal{X}}} [v(x) - \tilde{v}(x,r)]^2,$$

with $\tilde{\mathcal{X}} \subseteq \mathcal{X}$. We prefer to omit more involved structures (for example with products of variables), in order to prevent the complexity from increasing and thereby resulting in very long computation times. We decided to consider more elaborate structures only if results were not satisfactory. The experiments presented in the next section show that the structure from Equation (5.3) is already very useful, and that the fit is already accurate if we take $M = 2$. We remark that this is remarkable because we expect that the real value function of such systems also contains products of different variables.

Alternatively, the variables x_m can be split into two parts: the queue length and the number of specialists that are busy. This increases the number of basis variables from $2MK$ to $3MK$. Our experience is that, in conjunction with approximate dynamic programming, it does not yield significantly better results. More importantly, the computation times of approximate dynamic programming increase because more coefficients need to be determined.

Example

In this section we elaborate on an experiment that we executed in order to find an accurate structure for the value function. Our example concerns a two-skill call center with parameters $w_1 = 3, w_2 = 1, \lambda_1 = 1.5, \lambda_2 = 1.5, \mu_1 = 2, \mu_2 = 2, \bar{\mu}_1 = 0.25, \bar{\mu}_2 = 1, s_1 = 1, s_2 = 1, \bar{s} = 2$.

To explore the structure of the value function, we calculate the value function of this example numerically. This is achieved by means of the power series algorithm, see Chapter 4. We obtain under the initial policy, which gives higher priority to specialists, an average holding cost of 9.24. After one-step policy improvement, see Section 5.3, the average holding cost decreases to 7.89. We are interested in an approximation of the value function, such that the same improvement is obtained by one-step policy improvement.

To obtain the approximation as is given in Equation (5.3), the unknown coefficients r of $\tilde{v}$ need to be estimated. We approximate the value function with $\tilde{v}$ by minimizing the sum of the mean-square errors between $v(x)$ and $\tilde{v}(x)$ and applying one-step policy improvement to the approximation of the value function. With regard to the fit, we consider the states up to level $L = 10$, and only consider linear and quadratic terms ($K = 2$). This yielded an average cost of 7.89. Moreover, the states up to level $L = 5$ yields 7.89, up to level 3 yields 7.90 and up to level 2 yields 7.89. Thus, there is hardly any loss of performance as the number of states decreases.

More results are presented in Table 5.1. The table consists of three parts per instance. In the first place, the optimal r. In the second place, the average cost after one-step policy improvement, denoted by $\tilde{g}^1$. In the third place, the table presents the mean square error (MSE). In addition, the value in the brackets denotes the maximum of the value function within the state space up to state level L. It is included to compute the accuracy of the relative differences between the fit and the exact values.

From the examples we conclude that the influence of adding a constant to the structure from Equation (5.3) on the results is small. The MSE decreases, but the average cost after one step of policy improvement stays almost the same. For simplicity, and to reduce the computation times, we decided to ignore the constant, or intercept, in the next experiments.

By taking $K = 3$ no better results are found, and by taking $K = 1$ the best outcome is 9.15. From this example and others we conclude that $K = 2$ is sufficiently high for applying successfully one step of policy improvement with approximate dynamic programming.

If we fit $\tilde{v}$ only in states from even levels and truncate the state space at level 14, the improved system has an average cost of 8.24. This shows the high sensitivity to the choice of representative states.

Next, we approximate the value function by minimizing the Bellman error, instead of fitting the structure from Equation (5.3) to the real value function. The advantage is that the real value function is not required in the

Parameters	$L = 10$	$L = 5$	$L = 3$	$L = 2$
$(r_1^{(1)}, r_1^{(2)})$	(4.40;1.27)	(4.47;1.15)	(4.21;1.20)	(4.40;1.10)
$(r_2^{(1)}), r_2^{(2)})$	(0.21;0.29)	(0.27;0.28)	(0.56;0.17)	(0.44;0.20)
$(\bar{r}_1^{(1)}, \bar{r}_1^{(2)})$	(12.97;0.45)	(11.32;0.09)	(11.08;0.01)	(11.07;0.04)
$(\bar{r}_2^{(1)}, \bar{r}_2^{(2)})$	(-0.22;0.45)	(0.70;0.03)	(0.80;-0.08)	(0.69;0.01)
$\tilde{g}^1$	7.89	7.89	7.90	7.89
MSE	5.28 (168)	0.29 (50)	0.06 (27)	0.0013 (22)

Table 5.1: Coefficients without intercept ($K = 2$)

	$L = 5$	$L = 10$	$L = 15$	$L = 25$
$(r_1^{(1)}, r_1^{(2)})$	(2.25;0.71)	(4.24;1.02)	(5.46;1.01)	(7.13;1.11)
$(r_2^{(1)}, r_2^{(2)})$	(0.84;-0.07)	(0.78;0.18)	(1.04;0.17)	(1.52;0.20)
$(\bar{r}_1^{(1)}, \bar{r}_1^{(2)})$	(4.68;-0.09)	(10.43;-0.15)	(13.73;-0.17)	(20.29;-0.05)
$(\bar{r}_2^{(1)}, \bar{r}_2^{(2)})$	(1.25;-0.06)	(0.71;-0.16)	(0.25;-0.26)	(-1.02;-0.49)
$\tilde{g}^1$	9.90	7.89	7.88	8.27
MSE 1-5	110.7	3.4	8.3	121.7
MSE 6-10	993.0	95.0	15.0	181.9
MSE 11-15	5496.2	875.8	382.4	79.9
MSE 16-20	19704.8	4041.2	2533.6	264.3

Table 5.2: Coefficients with intercept ($K = 2$)

calculations, by which the curse of dimensionality is avoided. As a result, the algorithms will probably be scalable to larger call centers.

To investigate the sensitivity to the choice of representative states in more detail, we minimize the Bellman error over different sets of states. The sets are composed by truncating the state space at different levels. The results are presented in Table 5.2. Note the substantial differences in MSE between both tables. An important observation is that the average costs in Table 5.2 are much higher than in Table 5.1. We conclude that the performance of approximate dynamic programming highly depends on the composition of the set of representatives states. Even if the fit looks reasonably accurate, the policies resulting from the policy improvement method can be disappointing.

Parameters	1	2	3
$(r_1^{(1)}, r_1^{(2)})$	3.90,1.31	4.50,1.24	4.45,1.27
$(r_2^{(1)}), r_2^{(2)})$	-0.29,0.34	0.31,0.25	0.25,0.29
$(\bar{r}_1^{(1)}, \bar{r}_1^{(2)})$	13.04,0.41	13.14,0.42	11.96,0.91
$(\bar{r}_2^{(1)}, \bar{r}_2^{(2)})$	-0.15,0.41	-0.05,0.41	-1.23,0.91
r_1	0.16	-0.67	-0.21
r_2	0.00	0.08	0.55
r_3	0.00	0.10	0.56
r_4	0.00	-0.01	0.46

Table 5.3: Coefficients of experiments with product terms

Products of variables

We evaluated if products and powers of different variables can contribute to the performance of the policy improvements. Therefore, we added additional terms to the basis function from Equation (5.3). In our experiments we consider a two-skill call center. The call center has the same parameters as we used for Table 5.1. Moreover, the accuracy of the experiments is measured by considering the parameters $K = 2$ and $L = 10$, which corresponds to the first column of the table.

First we add the term $r_1(x_1 + \bar{x}_1)(x_2 + \bar{x}_2)$ to the basis function. This experiment is denoted by 1. Fitting this structure to the real value function yields an MSE of 4.65, which is only 10% lower than the original value of 5.28 from Table 5.1. In experiment 2 we add the terms $r_1(x_1+\bar{x}_1)(x_2+\bar{x}_2)+r_2(x_1+\bar{x}_1)^2(x_2+\bar{x}_2)+r_3(x_1+\bar{x}_1)(x_2+\bar{x}_2)^2+r_4(x_1+\bar{x}_1)^2(x_2+\bar{x}_2)^2$. This yields an MSE of 4.1, which is 20% lower than the original value. In experiment 3 we add the terms $r_1\bar{x}_1\bar{x}_2 + r_2\bar{x}_1^2\bar{x}_2 + r_3\bar{x}_1\bar{x}_2^2 + r_4\bar{x}_1^2\bar{x}_2^2$. The MSE is this time 5.19, which is not significantly lower than 5.28. The optimal coefficients of the experiments are presented in Table 5.3. These experiments indicate that the gain of using product terms is moderate. However, it remains as a subject for future research to find out how these structures behave in conjunction with the techniques that we mentioned in Section 5.4.

5.6 Algorithm

In this section we treat the way in which we applied approximate dynamic programming to our call center model. We first discuss methods to compose the set of representative states and then give an outline of the algorithm.

Simulating the set of representative states

As we mentioned before there are infinitely many states. Even if we truncate the state space at a certain level, the number of states is huge. An advantage of approximate dynamic programming is that it does not require that the sum of $D(x, r)$ is minimized over all states. We conjecture from the experiments in Section 5.5 that, in order to obtain good results, we can choose a smaller set of states, denoted by $\tilde{\mathcal{X}}$. This idea, in conjunction with approximate dynamic programming, is further analyzed in Section 5.7, by means of numerical experiments. The experiments reveal that taking a subset of states hardly results in a loss of accuracy. Nevertheless, the empirical composition of representative states is of crucial importance to the performance of approximate dynamic programming.

We generate the set of representative states by means of simulation, and in two different ways:

Sample path: A sample path is generated by means of simulation, consisting of q events, either arrivals or service completions. Thus, with a sample path we denote a random evolution of the state during some time interval. We include a warming-up period that is sufficiently long. The states that are traversed during the sample path are considered to be included in the set of representative states. Every state is picked according to a Bernoulli distributed variable with parameter p. If the variable takes 0, the state is ignored. If it takes 1, the state is included (such that the Bellman error is calculated for that state during the algorithm). The set of states is generated in advance and is kept fixed during the approximate dynamic programming algorithm. We denote approximate dynamic programming in combination with this method as ADP1.

Optimized: The set of states is composed iteratively such that after applying approximate dynamic programming and policy improvement a good policy is obtained. The details are given next. Initially, a very small set

of representative states is composed randomly, containing only four or five elements. This set is optimized by removing and adding states iteratively, while keeping the total number of elements the same. To determine the state to be removed we minimize the Bellman error of the remaining states and evaluate the one-step improved policy by simulation. The addition of a new state, which replaces the removed state, occurs similarly; the addition of a number of states is evaluated and the best state is selected. States are generated by means of simulation. The initial policy in each iteration is the best policy that has been found so far. We refer to this method as ADP2.

Because the empirical composition of the set of representative states is crucial for the quality of the resulting policies, we apply ADP1 several times. Moreover, a property of the system is that states from either high levels or low levels are not frequently visited during the simulation runs. Recall that the level of a state is defined as the total number of calls in the system. We observed that the approximations are less accurate for high- and low-level states, because one-step policy improvement often yielded bad actions for these states. Therefore, we apply policy improvements only to states in a certain range of levels. We denote $\underline{L}$ as the lowest level of states to which policy improvement is applied and the highest level will be denoted by L.

Computation scheme

In this section we discuss the approximate dynamic programming algorithms, called ADP1 and ADP2, which are straightforward to compose. Comparable schemes can also be found in most papers of the literature about this subject.

The algorithms yield a one-step improved policy. ADP1 is treated first, consisting of the following steps:

1. Choose the initial policy and generate the set of representative states, as described in Section 5.6. Suitable parameters are given in Section 5.7.

2. Determine the average cost by means of simulation.

3. Apply approximate dynamic programming conform Section 5.4.1.

4. Execute one-step policy improvement, see Section 5.3. Go to step 1 if the outcome is not satisfactory.

5. Optionally, fine-tune $\underline{L}$ and L afterwards by simulation and try different weights for the states by changing ρ such that the average cost is further reduced.

ADP2 integrates ADP1 with a method to optimize the set of representative states, which is achieved by local search. Details have already been given in this section. The main difference with ADP1 is an extra loop in order to optimize this set.

5.7 Numerical examples

In this section we discuss the numerical examples, categorized into two- and three-skilled call centers. These are solved by minimizing the Bellman error and using the gradient method that we described in Section 5.4.1. Finally, we present results that we obtained with approximate linear programming.

In Table 5.4 results are presented for six instances with two skills and three agent groups. Each column corresponds to one call center instance. The rows are grouped together: the upper block contains the model parameters and the lower part presents the performance measures. If we look at the second column from the left (instance 1), we read that under the initial policy the average cost is 7.8, against 3.6 under the optimal policy. By using ADP1 we obtain a policy with an average of 4.1 per unit of time, and with ADP2 an average of 3.7.

Table 5.5 shows the results of three call centers, with three skills and four agent groups, $M = 3$. Compared to the initial policy, approximate dynamic programming reduces the average cost by almost 50%. It is not possible to obtain and evaluate the optimal policies, because the state spaces of these examples are too large.

Some parameters are kept constant. With value iteration the state space is truncated at level 125. Concerning the gradient method, the initial step size γ is 0.2 and the scalar is 1.15^{-1}. Parameter K is fixed at 2. With ADP1 we take $p = 5.0 \times 10^{-5}$, which is the parameter of the Bernoulli distribution, and $q = 2.5 \times 10^{6}$.

Several runs are executed for each example, until the long-run average cost is about 50% of the average cost that corresponds to the initial policy. Finding a good stopping criterion is a subject for further research. Our experience is that the outcomes of the different runs vary substantially and are often even worse than under the initial policy. On average about ten runs

	Examples					
	1	2	3	4	5	6
w_1	0.5	2	2	2	0.5	0.5
w_2	2	0.5	0.5	0.2	2	2
λ_1	2.5	1.5	1.8	1.6	4.3	7
λ_2	2.5	1.5	1.8	1.6	4.3	7
μ_1	1	0.5	0.25	0.25	0.5	0.5
μ_2	1	0.5	0.25	0.25	0.5	0.5
$\bar{\mu}_1$	0.25	0.4	0.3	0.3	0.13	0.13
$\bar{\mu}_2$	1.8	0.6	0.2	0.2	0.38	0.38
s_1	2	1	2	6	7	11
s_2	2	1	2	6	7	11
$\bar{s}$	4	6	12	4	14	25
g^0	7.8	1.89	7.4	2.20	5.6	4.7
$\tilde{g}^1$ (adp1)	4.1	1.17	3.8	1.38	3.5	3.0
$\tilde{g}^1$ (adp2)	3.7	1.16	3.8	1.30	3.2	2.4
g (opt)	3.6	1.15	3.6	1.18	2.9	> 2.2

Table 5.4: Examples with two skills

of ADP1 were necessary to obtain the results presented in the table. The computation times are several minutes per instance, consisting of multiple runs.

We also applied ALP to the instances of our numerical experiments, but we were not able to obtain good routing policies. The average costs of the improved policies, obtained by means of simulation, were high. For the six cases from Table 5.5 the best results that we obtained, are average costs of 5.8, 1.72, 7.4, 1.63, 5.6, and 3.7, respectively. Because of unsatisfactory results we also considered the formulation with discounted costs, see Kallenberg (2002). We included a discount factor because, according to the literature, it can improve the quality of the outcomes significantly, see for example Farias and Van Roy (2006). This was an unsuccessful attempt because the policies did not perform better with inclusion of a discount factor.

5.8 Concluding remarks

Standard methods for analyzing multi-skill call centers suffer from the curse of dimensionality. This chapter considers approximate dynamic program-

	Examples		
	1	2	3
(w_1, w_2, w_3)	(0.5,2,0.5)	(0.5,2,0.5)	(2,0.2,1)
$(\lambda_1, \lambda_2, \lambda_3)$	(2.5,2.5,2.5)	(4.3,4.3,4.3)	(1.6,1.6,1.6)
(μ_1, μ_2, μ_3)	(0.5,0.5,0.5)	(0.5,0.5,0.5)	(0.25,0.25,0.25)
$(\bar{\mu}_1, \bar{\mu}_2, \bar{\mu}_3)$	(0.13,0.9,0.13)	(0.13,0.38,0.13)	(0.3,0.2,0.15)
(s_1, s_2, s_3)	(4,4,4)	(7,7,7)	(6,5,4)
$\bar{s}$	12	21	8
g^0	14.0	10.5	14.7
$\tilde{g}^1$ (adp1)	7.0	5.5	8.1
$\tilde{g}^1$ (adp2)	6.6	5.2	8.5

Table 5.5: Examples with three skills

ming. In contrast to dynamic programming, the curse of dimensionality is irrelevant to approximate dynamic programming. We obtain dynamic policies for larger call centers than is possible by the standard techniques from the literature, discussed in Chapter 4.

Approximate dynamic programming requires a structure of the value function, which is supposed to approximate the real value function. According to our experiments, using basis functions of order two, see Section 5.5, is sufficient to obtain nearly optimal policies numerically. We evaluated this by fitting second-order polynomials to the real value function. However, according to our experience, the minimization of the Bellman error yields approximations that are less accurate than the ones obtained by using the real value functions, which results in bad policies. This indicates that there exist second-order polynomials that approach the real value function closer than the ones we obtained with approximate dynamic programming, which shows that there is scope for further improvements.

We found that the choice or composition of representative states is crucial for finding good policies. From our experiments we conclude that generating sample paths by means of simulation is an effective way to compose the set of representative states. We run the algorithms several times, with different sets of representative states, because the policies are very sensitive to the composition of the set of representative states.

It is in our opinion interesting for further research to understand the relation between the choice of representative states and the accuracy of the fit better. This could eventually result in a smaller set of representative

states, and reduce the computation times significantly, by which it would become easier to apply approximate dynamic programming to bigger call centers.

In future research we aim to extend the analysis to call centers with more skills and additional agent groups. There are several issues that need to be solved in advance, and require further research. In the first place, the computation times depend to some extent on the type of gradient method. It seems to be worth studying the literature and to find more effective methods, such that the computation times are reduced. This is required for the analysis of larger call centers. In the second place, another direction is to improve the structure of the basis function, and maybe to include products of different variables, or to change the definition of the state such that the variables have another meaning. Finally, the state space could be divided into multiple regions and separate functions could be fitted to each region, see for example Poupart, Patrascu, and Schuurmans (2002).

Chapter 6

Adaptive Control of Routing Policies

This chapter introduces a policy to control the service level in multi-skill call centers. The call center has agents that handle multiple types of jobs, as well as specialized agents. The service level is controlled by a policy that assigns jobs to the flexible agents. This policy is changed dynamically, while assignments to the specialists occur according to a fixed rule. The dynamic policy allows us to divide the agents' capacity among the different job types, continuously in time, such that the service levels among the different job types can be optimized. It is an online method because the decisions are based upon the real-time situation.

An advantage of the control policy is that it requires no information about the arrival processes in advance. If fluctuations in workload occur, the policy is adapted such that targets concerning the service levels are met.

6.1 Introduction

This chapter deals with performance optimization, and our objective is to meet different targets concerning service-level measures. Several customer classes are distinguished and we aim to provide a higher service level to more important customers. We assume that the scheduling of agents has already taken place, such that the number of available agents during the day is known in advance.

The service levels are influenced by the way in which calls are assigned to the agents. In practice, routing policies often take the number of customers

in the system into account, but do not respond to the service levels that are provided in the preceding period. In this chapter, we consider several optimization techniques that take the real-time service levels into account and, hence, can be classified as online updating methods and real-time routing. The word "online" indicates that adjustments to the policy are made during the evolution of the process. In our application, the adjustments depend on the history of the process. Moreover, in this chapter we consider policies that:

- are simple,
- are easy to generalize to a large call center with different agent groups, and
- use no specific information about the call center, and in particular no parameter values.

An important reason why online adjustments are important in call centers, is given next. To calculate staffing levels and schedules, call centers use predictions of the expected future workload. Bad predictions usually have a negative impact on the service levels or the productivity. If the amount of work offered to the agents deviates from their service capacity, either customers have to wait or agents are idle, resulting in low service levels or low productivity. Several studies have shown that the arrival process and the workload are hard to predict in call centers, see for example Jongbloed and Koole (2001), Avramidis, Deslauriers, and L'Ecuyer (2004). This makes online optimization important.

In multi-skill call centers, we can expect that predictions of a certain call type have a relatively lower accuracy than predictions of the total amount of work. Recall that smaller numbers are relatively more difficult to predict, because the standard deviation is relatively larger than the expectation. We refer to Section 1.3 for a more extensive explanation about the predictability of workload. Obviously, inaccurate predictions can result in a mismatch between workforce and workload per job type. For that reason, routing policies that do not depend on the workload can be suboptimal in multi-skill call centers with regard to certain objectives. In particular, if a routing policy would not be updated during the day, we can show that the service level is driven by the amount of work that arrives of each type. The next example shows that, as a result, important customers can get on average a lower service level than less important customers.

Example: Consider a two-skill call center with exponentially distributed service times with an average of 5 minutes, 5 specialists of both types, 40 generalists, and customers having an average patience of 5 minutes, which is also exponentially distributed, see for example Figure 4.1. Jobs are assigned to generalists if all specialists are busy, and the policy is work-conserving. The jobs arrive according to Poisson processes. If the arriving workload of both types is on average 5 per minute, 5.6% of the customers of each type abandons the queue, such that the ratio of the service levels is 5.6/5.6=1. However, if the workload turns out to be on average 4 and 6 calls per minute for types 1 and 2, the abandonment rates are 5.97% and 5.14%, respectively. These percentages correspond to a ratio of 5.97/5.14=1.16, which differs from the intended ratio of 1. If the customers of type 1 are more important, this situation is undesired. We remark that without specialists the ratio would still be 1, see Theorem 6.2.

This kind of scenario can often be avoided by making adjustments to the routing policies. We show in this chapter that applying online optimization to routing policies is an effective way to compensate workforce for deviations between the workforce and the workload among the different job types, and to meet targets concerning service levels.

We restrict the optimization to call-selection policies. A call selection takes place when an agent finishes a call. Then one has to decide from which queue the next job is taken. In multi-skill call centers this is an important decision because the availability of agents and queue lengths have an impact on the service levels. In addition, we assume a static policy for the assignment of calls to agents. These event occurs at the arrival epochs of calls when there is at least one agent available with the right skill, also referred to as an agent-selection policy. The main reason for optimizing only the call-selection policy is that, according to our experience, the impact on the performance is higher compared to agent-selection policies. This holds especially if the queues are non-empty during a high fraction of time, which is more likely to occur in larger call centers. We remark that the impact of call-selection policies is always expected to be low if the workload is relatively low (since the queues are during a high fraction of time empty). However, this results in high service levels, such that meeting targets on service-level measures is less important.

Moreover, non-work-conserving policies, such as policies using thresholds

or having reservations of agents for important job types, are not considered in this chapter. In our opinion, the restriction does not reduce the usefulness of the method because we conjecture that the larger the call centers, the more effective work-conserving policies are, see Example 2 in Section 6.3.3. A reason is that in larger call centers the number of calls in the queues is on average higher and more equally divided among the job types, such that the response of the control policy is higher. In addition, an advantage of work-conserving policies is that, in certain cases, work-conserving policies minimize the abandonment rate, see Theorem 6.1. Another reason is that we want to prevent the analysis and the control policy from being too complicated.

Literature

Call-selection policies can be classified in static and dynamic policies. With dynamic policies the state of the system is taken into account, see Section 3.4.1 for an explanation. Optimal policies are most often dynamic policies. Structural results about optimal policies are, for example, given in Örmeci (2004), Guérin (1998), Schaack and Larson (1986), and Peköz (2002), see Section 3.6 for discussions. Static policies have already been discussed in Section 3.7.3.

We remark that the papers from the literature most often consider arrival processes that are more predictable than in reality, namely Poisson processes with a fixed parameter. Moreover, they illustrate that optimal results are difficult to obtain. The structures of optimal policies are model-dependent and often difficult to generalize to more complicated models.

The literature presents different numerical techniques to improve the performance of stochastic processes. Examples are perturbation analysis and likeli-hood-ratio methods, see for example Meketon (1987) for a survey. Both methods have in common that they allow us to update the policy in an online fashion. They can also handle arrival processes of which little is known in advance, as is the case in our study.

For more recent papers on stochastic optimization we refer to Swisher and Hyden (2000) and Azadivar (1999), presenting a survey of relevant techniques, and Maryak and Chin (2001), who study methods to determine a global optimum.

Content

This chapter is structured as follows. Section 6.2 describes the model, introduces the control policy, and presents structural results. Methods to update the parameters of the control policy are discussed in Section 6.3. The methods are applied to a number of cases in Section 6.4. Finally, our conclusions are summarized in Section 6.5.

6.2 Framework

The control mechanism that is presented in this chapter has different types of applications. In this chapter it is applied to a call center with M skills, having specialists and fully cross-trained generalists. The model can be classified as a multi-skill $GI/G/s + M$ system; inter-arrival times and service times are allowed to be generally distributed. As a special case we discuss the model in the case of two skills. A description of the general model is given first.

6.2.1 Model

We consider a call center with M different arrival processes that contain different types of jobs, and each process is denoted by an index m, with $m \in \{1, 2, \ldots, M\} =: \mathcal{M}$. The call center contains M groups of specialists, with group indices $1, 2, \ldots, M$, and one group of generalists with index 0. According to the notation of Chapter 2 we have $\{0, 1, 2, \ldots, M\} =: \mathcal{G}$. The skill set of specialists from group g is defined as $S_g = \{g\}$. Generalists have all skills such that $S_0 = \mathcal{M}$. We assume that the agents from the same group behave statistically identically with respect to the service times. All traffic is inbound and we make no assumptions on the arrival process. The inter-arrival times have, for example, general, time-inhomogeneous distributions. A graphical representation of the model is depicted in Figure 5.1.

The service times have a general distribution. However, in the examples of Section 6.4, the service of a job requires an exponentially distributed amount of time and is job-type and group-dependent, with rates μ_m for type-m specialists, and generalists work with rate $\bar{\mu}_m$ on type-m jobs. Each job is served by exactly one agent, either a specialist or a generalist. At arrival, a job is assigned to a specialist if one is available, and otherwise to a generalist. If a customer enters the system when all agents that can handle the call are busy, he or she has to wait in the queue associated with

the call type. At a service completion the next job is determined according to waiting time factors, which will be explained later in this section. The system is work-conserving, i.e., jobs only wait if all servers are busy.

Customers in each queue have patience for staying in the queue. We assume that all customers with the same type of call have the same distribution of patience, which is the exponential distribution with rate γ_m associated with customers of type m. A customer abandons the queue as soon as the waiting time exceeds his or her patience.

Let us assume that the system evolves during a period $(0, T)$ under a certain control policy, being defined in Section 6.3, and n_m customers of type m are served during the interval. We define $a_m(j)$ as the indicator denoting if the j-th served customer of type m received a bad service, which we will define either by an abandonment or a waiting time longer than a certain value, and $b_m(j)$ denotes the waiting time of the customer, with $j = 1, 2, \ldots, n_m$. If a customer abandons the queue, without being served, its waiting time is defined to be zero.

6.2.2 Objective

The system is controlled by a work-conserving non-preemptive call-selection policy π, which will be defined in more detail later in this section.

Let us define the service level of type-m customers under policy π as the fraction of customers whose experienced service quality is below a certain criterion. The service level is measured either by customers who abandon due to impatience or customers that wait longer than a certain acceptable value. These services will both be considered as bad. The service level of type m is denoted by $Q(a_m)$, with a_m defined in Section 6.2.1. To simplify notation, the symbol π is not added to the variables, i.e., as a subscript or superscript. However, all statistics, such as service levels and waiting times, depend on π. The service level is calculated by

$$Q(a_m) = \frac{\sum_{j=1}^{n_m} a_m(j)}{n_m}. \tag{6.1}$$

Our objective is to find a control method that optimizes the routing policy, such that customers with a bad service have a certain distribution among the job types. An example of such a control method is giving high-priority customers a service level that is twice as high as the service level that low-priority customers receive, in case of two job types. Thus, this enables a

call center to offer more important customers a higher service level than less important customers. The desired distribution of the customers with a bad service can be specified by a parameter c_m for each type $m \in \mathcal{M}$, such that the policy yields service levels that are distributed proportionally to the parameters. For example, given c_a and c_b, we are interested in a policy that yields service levels of type a and b proportionally to c_a and c_b. We translate this to the objective function

$$\min_{\pi \in \Pi} \max_{m \in \mathcal{M}} \frac{Q(a_m)}{c_m}, \tag{6.2}$$

with Π the set of all routing policies that satisfy the conditions from Section 6.1. Finally, we also require the control method to minimize the second moments of performance measures, in particular the standard deviations of the empirical waiting-time distributions. The reason is that the average abandonment rate is in certain systems independent of the routing policy, such that several policies can satisfy (6.2), see Theorem 6.2. Recall, we restrict policy π to be an online optimization rule that only uses historical data, uses as little model-specific information as possible, and is easy to implement.

Class of policies

Jobs from the waiting queues are selected by using so-called waiting time factors, as described in, for example, Lu and Squillante (2004). These selections occur after each service completion of an agent, when there is at least one non-empty queue. The choice about the next call that is served depends on waiting time factors, of which exactly one is associated with each queue. When selecting a job, the agent considers the first job in each queue. From this set of jobs, he chooses the job of which the product of the waiting time and the waiting time factor is the highest. The factor is queue-dependent only.

An advantage of waiting time factors is their flexibility with regard to the many different routing policies that are possible. Consider for example the case that $M = 2$. Setting both waiting times equal results in a policy that serves jobs from both classes in a first-come-first-serve order, and by taking one waiting time factor equal to 0, one of the types has full priority above the other, independent of the waiting times. Waiting time factors between 0 and 1 appear, according to our experiments, to give system performance between the two cases with full priority to one of both types.

6.2.3 Structural results

A motivation for considering only work-conserving policies is given by means of Theorem 6.1. The theorem concerns a $G/M/s + M$ system, which has a generally distributed arrival process, exponential service times, s servers, and exponentially distributed patience times. In this chapter we assume that service times are not known before a service starts.

Theorem 6.1. *Consider a $G/M/s + M$ system with non-preemptive non-anticipating service discipline. The average abandonment rate is equal or higher under non-work-conserving policies than under work-conserving policies.*

Proof. We prove the theorem by considering two systems with different service policies. These are two identical $G/M/s + M$ systems (except for the service policy), denoted by 1 and 2. System 1 has a work-conserving policy and system 2 has a non-work-conserving policy. We allow servers in system 2 to be idle. All servers in system 1 are busy as long as jobs are present.

We describe a method to construct a sample path in both systems during a period of length T, and the systems start in the same state. The method can generate all possible sample paths in each system, according to the right distributions. We have to show that for a coupled sample path in both systems the number of abandonments is higher, or equal, in system 2. This will imply a higher or equal expected abandonment rate in system 2 than in system 1 when T goes to infinity.

The construction of the sample paths is as follows. The inter-arrival times, remaining patience times, and remaining service times are coupled between both systems. All arrival epochs coincide in both systems. Because patience and service times are exponentially distributed, they are memoryless. This allows us to assign new times to the jobs after each event. We describe a procedure that assigns exactly one time to each job after each event, either a service time to a job that is being served or a patience time to a job that is waiting in the queue. A time is generated randomly from the corresponding distribution and each time is assigned twice: to a job in system 1 and to a job in system 2. The assignment repeats, and it continues until all customers have received a new time, unless the number of customers in service or in queue is higher in one of both systems. In that case, times are assigned to the remaining jobs afterwards. Thus, not each time is assigned to a job in both systems. Consequently, if a time is assigned to a job in only

one of both systems, this time is not coupled to a job in the other system. As a result, the processes can differ between both systems if not all service and patience times are coupled.

For an arbitrary choice of coupled sample paths, we define $a_i(t)$ as the total number of abandonments until time t in system $i \in \{1, 2\}$, and we define $x_i(t)$ as the number of jobs in system i at time t, with $t \in [0, T]$. We show that for every t it holds that

$$\begin{aligned} a_1(t) &\leq a_2(t) \text{ and} \\ x_1(t) + a_1(t) &\leq x_2(t) + a_2(t). \end{aligned} \tag{6.3}$$

Assume that (6.3) holds until time epoch t at which an event occurs. We show that the inequalities also hold after the event. The symbol t^+ denotes a time epoch immediately after the event and t^- denotes a time epoch just before the event. To prove that (6.3) also holds after the event, we consider all possible events:

- There is a departure in both systems of jobs with coupled service times. Then (6.3) holds because we have

$$x_1(t^+) = x_1(t^-) - 1 \text{ and } x_2(t^+) = x_2(t^-) - 1.$$

- There is an abandonment in both systems of jobs with coupled patience times

$$\begin{aligned} x_1(t^+) &= x_1(t^-) - 1 \text{ and } x_2(t^+) = x_2(t^-) - 1, \\ a_1(t^+) &= a_1(t^-) + 1 \text{ and } a_2(t^+) = a_2(t^-) + 1. \end{aligned}$$

  Then (6.3) also holds.

- A departure occurs in system 1, such that

$$x_1(t^+) = x_1(t^-) - 1,$$

  which also satisfies (6.3).

- There is a departure in system 2

$$x_2(t^+) = x_2(t^-) - 1.$$

  This event is only possible if $x_1(t^-) < x_2(t^-)$, such that (6.3) holds at time epoch t^+.

- There is an abandonment in system 1

$$x_1(t^+) = x_1(t^-) - 1,$$
$$a_1(t^+) = a_1(t^-) + 1.$$

  This is only possible if $x_1(t^-) > x_2(t^-)$, which implies that $a_1(t^-) < a_2(t^-)$. We see that (6.3) is also satisfied after the event.

- Finally, if a job in system 2 abandons, we get

$$x_2(t^+) = x_2(t^-) - 1$$
$$a_2(t^+) = a_2(t^-) + 1,$$

  which also satisfies (6.3).

If we let T go to infinity, we obtain an abandonment rate that is equal or higher in system 2, just by dividing each $a_i(t)$ by the total expected number of arrivals, which is equal in both systems. □

It is intuitively clear that the theorem also holds for general service-time distributions, because the abandonment rate will be minimized by minimizing the amount of work in the queues.

Concerning work-conserving policies we emphasize a number of theorems that are relevant, of which a few originate from Jouini, Pot, Dallery, and Koole (2006). These theorems are presented next, in conjunction with an alternative proof than is given in the paper. In most cases, the service times are assumed to be mutually independent and generally distributed.

Theorem 6.2 (Jouini, Pot, Dallery, and Koole (2006)). *Consider a $G/GI/s+M$ system. Then the average abandonment rate over all customers are the same for any work-conserving non-preemptive non-anticipating routing policy.*

Proof. We prove the theorem by considering two systems with different work-conserving policies. We describe a method to construct a sample path in both systems during a period of length T. The method can generate all possible sample paths in each system. We show that the abandonment rates and average waiting times are equal in both systems, because the evolution of the systems do not depend on the policy. This implies an equal expected abandonment rate and average waiting time in both systems when T goes to infinity.

The sample paths are constructed as follows. Consider two identical $G/GI/s+M$ systems, say system 1 and system 2, having different policies for the assignment of calls to agents. Our approach is to create in both systems the same arrival times and also couple the successive service times in both systems, i.e., the k-th service time is equal in system 1 and 2, for all k, and is independent of the type of the k-th job in each system. The times are generated randomly from the distributions. We are allowed to choose a service time at the start of a service, instead of at the arrival epochs, because the service policy is non-anticipating.

We consider the evolution of the process. During the process three types of events can occur: arrivals, service completions, and abandonments. It is clear that all three types of events coincide, because arrivals occur at the same time epochs by the coupling, service completions occur at the same time because the service times are appropriately coupled, and abandonments occur at the same time epochs because they can always be coupled between both systems due to the exponential distribution, which is memoryless.

Thus, the process of the total number of jobs in both systems is identical for both systems and the abandonment rates are equal. □

Moreover, we also found a relation between the average waiting times for any work-conserving non-preemptive routing policy.

Corollary 6.3. *Consider a $G/GI/s+M$ system. Then the average waiting time over all customers are identical for any work-conserving non-preemptive non-anticipating call-selection policy.*

Proof. Consider the proof of Theorem 6.2 and notice that the cumulative waiting times increase equally in both systems. Thus, the waiting times of all customers, including the waiting times of the abandoned customers, coincide in both systems at each point in time. □

The next example shows that the average waiting times of the served customers (excluding the waiting times of the abandoned customers) depend on the service discipline. It also shows that if a customer with an infinitely long patience would enter the system, its expected waiting time depends on the service discipline as well.

Example: We consider a single-skill call center, an $M/M/s+M$ queue under non-preemptive work-conserving service disciplines. The arrival rate is 1

per minute, the service rate 0.2, the abandonment rate 0.4, and 7 agents are present. Under a first-come-first-serve service discipline the average waiting time over all customers is 10.0 seconds, while the average waiting time of the served customers is 7.6 seconds, and the average waiting time of customers with infinitely long patience would be 14.1 seconds. Under a last-come-first-serve service discipline the average waiting time over all customers is also 10.0 seconds, but the average waiting time of the served customers is 6.4 seconds and of the customers with an infinitely long patience 17.6 seconds.

Concerning this example, we have the following result, assuming that the first moments of the distributions are bounded.

Theorem 6.4. *Consider the $G/GI/s + M$ system under work-conserving non-anticipating service disciplines. The average waiting time of the served customers depends on the service discipline, and is minimized by the last-come-first-served (LCFS) discipline and maximized by the first-come-first-served (FCFS) discipline.*

Proof. Consider the two systems from the proof of Theorem 6.2, that are coupled similarly. In system 1 we apply LCFS, and in system 2 customers are served according to FCFS. The total waiting time of the served customers is counted in both systems, and increased at the beginning of each service by the waiting time of the customer that starts service. The sample paths show immediately that the total waiting is minimized by LCFS and maximized by FCFS, and the increase of the total waiting times are always at least higher in system 2. If there are at the beginning of a service at least two customers in the queue, the increase of the waiting times is strictly higher in system 2 than in system 1. Note, when assuming strictly positive and bounded parameters, there is always a positive probability that the system moves to a state with more than two customers in the queue. For that reason, the average waiting times of the served customers can be different in both systems, in the limit. □

Theorem 6.2 also holds in case of two customer classes, which can easily be extended to more classes.

Corollary 6.5 (Jouini, Pot, Dallery, and Koole (2006)). *Consider a $G/GI/s + M$ system with two classes of customers A and B. Then the total abandonment rate is the same for any work-conserving non-preemptive non-anticipating service discipline.*

Proof. This is an immediate consequence of Theorem 6.2. The arrival stream can be divided into two separate streams, without any restriction. □

We define c_π as the ratio of the abandonment rate of type 1 and type 2 under work-conserving policy π in the limit.

Corollary 6.6 (Jouini, Pot, Dallery, and Koole (2006)). *Consider a $G/GI/s + M$ system with two classes of customers A and B. Let π_A (π_B) be the policy that gives strict non-preemptive priority to customers A (B). Then for any work-conserving non-preemptive non-anticipating policy, π, the achieved ratio of the abandonment ratios in the stationary regime, c_π, satisfies the relation*

$$c_{\pi_A} \leq c_\pi \leq c_{\pi_B},$$

with c_{π_A} and c_{π_B} the achieved ratios of the abandonment rates in the stationary regime for π_A and π_B, respectively.

Proof. We prove the theorem by considering two $G/GI/s+M$ systems with different work-conserving policies, say 1 and 2. Firstly, we describe a method to construct a sample path in both systems such that the sample paths occur with equal probability. The method can generate all possible sample paths in each system. Next, we show that each construction of a sample path in both systems yields service levels that satisfy the above inequalities. This implies that the expected service levels also satisfy the conditions.

Both systems are coupled with respect to arrival epochs, service times, and remaining patience times, such that the epochs of arrivals, service completions, and abandonments coincide. This is achieved similarly to the previous proofs; the k-th service times of both systems are coupled, independent of the job type, abandonments are coupled because of the memoryless property. Hence, the total number of jobs is the same in both systems at any time, according to the proof of Theorem 6.2.

We consider the processes with randomly chosen service, abandonment, and inter-arrival times, according to the exponential distributions. In system 1 full priority is given to customers of type A, compared to system 2 in which this does not always occur. By always taking jobs from queue A if the queue is non-empty, as is the case in system 1, the queue length of A is minimized over time, and smaller than the queue length in system 2. The abandonment rate is smaller too because the remaining patience times of all type-A customers in system 2 can be coupled to jobs of type A in system 1. Thus, in system 1 the abandonment rate of type A divided by

the abandonment rate of type B is minimal, which will converge to c_A in the long run, according to the definition. A similar explanation holds for c_B. Finally, by replacing the service of type-A jobs by the service of type-B jobs, the queue length of type A increases and decreases for type B, while the processes of both systems are still coupled. Consequently, the empirical ratio of abandonment percentages increases. The replacement can be applied successively until customers of type B receive full priority. Thus, the two most extreme policies are π_A and π_B, and by replacement the ratio can increase from c_A to c_B in the long run. □

This result indicates that policies based on waiting time factors can yield performance that covers the whole range of work-conserving policies, also in case of more than two classes. Note that c_π is equal to 1 if the waiting time factors of both types are equal.

With the waiting time being defined as the waiting time of all customers, including served and abandoned customers, the following theorem holds.

Theorem 6.7. *Consider two queues with equally exponentially distributed patience times. Then the ratio between the abandonment rates in both systems and the ratio between the average waiting times are the same.*

Proof. Using Little's law we can see that if the average waiting time in both systems are equal, then the average queue length are equal too. Because the patience times are memoryless, the abandonment rates are proportionally to the queue lengths. □

As a result of this theorem, we conjecture that Theorem 6.6 also holds with regard to waiting times.

6.3 Updating mechanism

In this section we explain how the waiting time factors are changed over time. In case of two job types, we denote the factors at time $t \in (0, T)$ by $w_1(t)$ and $w_2(t)$, associated with types 1 and 2, respectively. The value of $w_m(t)$ at time t depends on the history of the process. The higher the waiting time factor of a queue, the earlier the customers from the associated call type are served. As a result, the waiting times get on average shorter and the number of customers with a bad service decreases, which improves the

service level. Meanwhile, the service levels of the other job types decrease because these jobs have to wait longer. Thus, the values of the factors have an impact on the service levels. By adjusting the factors $w_m(t)$, the ratio between the service levels can be directed upwards and downwards, such that the desired ratio can be reached if it is feasible within the class of work-conserving routing policies.

6.3.1 Method 1

We consider a control policy π that makes adjustments to the waiting time factors in an online fashion. The policy has one parameter, which we denote by β, with $\beta \in [0, 1]$. This parameter determines the possible combinations of the waiting time factors. In case of two job types, the waiting times factors can take two different combinations of values at each point in time, namely $(w_1(t), w_2(t)) \in \{(1, \beta), (\beta, 1)\}$. In case of more than two job types, say M, the waiting time factors at time t are denoted by vector $(w_1(t), w_2(t), \ldots, w_M(t)) =: w(t)$. The set of possible values of $w_m(t)$ that we allow is $\{\beta, \beta + 1\tau, \beta + 2\tau, \ldots, \beta + (M-1)\tau\} =: H$, with $\tau = \frac{1-\beta}{M-1}$. Consider for example $M = 4$ and $\beta = 0.4$, then $H = \{0.4, 0.6, 0.8, 1.0\}$. We restrict the vector of waiting time vectors $w(t)$ at some time t to be a permutation of H. So queues are not allowed to have identical waiting time factors. Optimizing the composition of H is a subject for further research. Additional information about the policy is given in Section 6.3. A discussion of the updating methods that we considered, is given in Section 6.3.3.

Updates of the waiting time factors occur as follows. After each event the service level of each type is estimated by considering the historical events. The estimates are calculated by dividing the number of observed bad services by the number of served customers. The time horizon of the history is specified for each numerical experiment separately. Next, we let these service levels determine whether or not the waiting time factors are changed. In case of two skills, if the ratio between the two service levels at time t exceeds c_1/c_2, the waiting time factors are interchanged, (w_1, w_2) is set to (w_2, w_1). Let us define $Q_1(t)$ and $Q_2(t)$ as the empirical service levels of both job types at time t, then we can write

$$(w_1(t), w_2(t)) \leftarrow \begin{cases} (1, \beta) \text{ if } c_2 Q_1(t) > c_1 Q_2(t) \\ (\beta, 1) \text{ if } c_2 Q_1(t) \leq c_1 Q_2(t) \end{cases} . \tag{6.4}$$

In case of more than two skills we count the total number of bad services

over all types. Then we divide the resulting number among the job types in such a way that the distribution of the percentages customers with a bad service is proportional to the parameters c_m. Next, we calculate for each type the difference between on the one hand the percentages resulting from this calculation and on the other hand the real bad-service percentage observed until time t. We order the differences increasingly and we order the elements in H increasingly. The assignment of the elements from H to the job types is determined by this ordering; the job type with the highest difference gets the highest waiting time factor, the job type with the second-highest difference gets the second-highest waiting time factor, etc.

6.3.2 Method 2

We consider a call center with two job types. A method for updating the waiting time factors is to keep w_1 fixed and to update w_2 dynamically. Under the assumption that w_1 is fixed, it is straightforward to decrease w_2 when Q_1 is high, and increase w_2 otherwise. A difficulty is to develop a method that updates w_2 in a simple but effective way, without many control parameters. We investigated several possibilities by executing experiments. As a result, the parameter w_2 is decreased by multiplication by η and increased by division by η, with $0 < \eta \leq 1$. The performance is discussed in Section 6.4. We decided to update w_2 after each event.

6.3.3 Discussions

A well-known method for stochastic approximations is presented in Robbins and Monro (1951). We discuss if our control mechanism fits in the framework that they consider. Hence, we describe their framework, which is achieved by giving a citation from the abstract of their paper: *"Let $M(x)$ denote the expected value at level x of the response to a certain experiment. $M(x)$ is assumed to be a monotone function of x but is unknown to the experimenter, and it is desired to find the solution $x = \theta$ of the equation $M(x) = \alpha$, where α is a given constant. We give a method for making successive experiments at levels $x_1, x_2, \ldots$ in such a way that x_n will tend to θ in probability."*

In order to relate their model to ours, we consider our model in case of two skills. Because the arrival rates fluctuate over time, the control problem is like $M(x, \lambda(t))$, compared to $M(x)$ in Robbins and Monro (1951), with x playing the role of $w_1(t)$ and $w_2(t)$, M denoting $Q_1(t)/Q_2(t)$, and $\lambda(t)$ cor-

responding to the arrival-rate function of each job type. Thus, the response function M depends on t by $\lambda(t)$ in our model. For that reason, $w_1(t)$ and $w_2(t)$ need to be responsive and should not converge over time. Whereas, x_n converges to θ in Robbins and Monro (1951). Hence, we conclude that, if arrival rates fluctuate over time, our control problem does not fall straightforwardly within their framework, such that the problem cannot easily be solved by their method.

In general, it holds that the higher the absolute difference in values between the waiting time factors, i.e., $\beta = 0$, the more reactive the policy is with regard to meeting the objectives. However, there is also a drawback; according to Lu and Squillante (2004) we can expect, in the case of a single server, that the variance of the waiting times increases. This is also intuitively clear; the low-priority jobs must sometimes wait for high-priority jobs while the jobs from the other types are served first, which increases the differences in waiting times.

We remark that the control policy does not anticipate on future events. It just reacts to the realization of the ratio that is determined by the history of the process.

In our studies we ignored strictly non-work-conserving policies because we conjecture that they are nearly optimal in call centers with many servers and realistic parameters. However, optimal routing sometimes requires idling. This has been shown for a multi-server queue in Yahalom and Mandelbaum (2005). We show the impact of idling by using the online tool located at our website
http://www.math.vu.nl/∼sapot/software/Sim2Skill. It contains an implementation of the model from this section in case $M = 2$, with waiting time factors being fixed over time. In addition, a reservation level can be specified, which denotes the number of generalists that is reserved for type-1 calls. An arriving call of type 2 is only assigned to a generalist if the number of available generalists is higher than the reservation level.

Example 1: Consider the call center with arrival rates of 2 per minute: $\lambda_1 = \lambda_2 = 2$, service rates of 5 minutes: $\mu_1 = \mu_2 = \bar{\mu}_1 = \bar{\mu}_2 = 0.2$, abandonment rates of 5 minutes: $\gamma_1 = \gamma_2 = 0.2$, group sizes of $s_1 = s_2 = 5$ and $\bar{s} = 13$, and AWTs of 20 and 30 seconds, respectively. Our objective is that the number of customers of type 2 with a waiting time longer than 20 seconds is 2.5 times higher than for type 1, i.e., $c_1 = 1$ and $c_2 = 2.5$. Using waiting times of $w_1(t) = w_2(t) = 1$ yields 17 and 13 percent of customers whose

waiting time is longer than the AWT for types 1 and 2, respectively. Next, we take $w_1(t) = 1$ and $w_2(t) = 0$, which yields 13 and 15 percent. Setting the reservation level to 1 gives the desired ratio of 2.5, with percentages of 8 and 20, corresponding to 92 and 80 percent customers who wait shorter than the AWT. If the empirical ratio of service levels does not converge exactly to 2.5 for a certain reservation level, we propose to randomize the reservation level. This is a subject for future research.

Moreover, we conjecture that strictly non-work-conserving policies are only important to relatively small call centers. We refer to Section 6.1 for an intuitive explanation. Next, an illustration is given by means of an example. We consider a system under a work-conserving policy. The example shows that a ratio between service levels is more easily reached if the call center is larger.

Example 2: Call centers of three different sizes are compared under fixed priority routing, i.e., $\beta = 0$. We show that the ratio between the service levels increases when the call center grows. This suggests that, to obtain the same ratio for larger sizes, β can be increased. Consider a call center with parameters $\mu_1 = \mu_2 = \bar{\mu}_1 = \bar{\mu}_2 = 0.2$, $\gamma_1 = \gamma_2 = 0.2$, and waiting time factors $w_1(t) = 1$ and $w_2(t) = 0$. We consider three situations (with almost equal service levels if β would be 1):

1. $\lambda_1 = \lambda_2 = 1$, $s_1 = s_2 = 1$, $\bar{s} = 9$, yielding $Q(a_1)/Q(a_2) = 0.5$,
2. $\lambda_1 = \lambda_2 = 5$, $s_1 = s_2 = 5$, $\bar{s} = 40$, yielding $Q(a_1)/Q(a_2) = 0.2$, and
3. $\lambda_1 = \lambda_2 = 50$, $s_1 = s_2 = 40$, $\bar{s} = 400$, yielding $Q(a_1)/Q(a_2) = 0.05$.

Thus, the ratio becomes smaller when the size of the call centers increases. In Section 6.4 we will show by means of examples that when β increases, the variance in the waiting times decreases.

Perturbation analysis

Examples of other optimization methods than the one presented by Robbins and Monro (1951), are perturbation analysis and likelihood-ratio methods. We refer to Meketon (1987) for references on perturbation analysis.

Our objection to these methods in general, is that they require detailed information about the system. For example, perturbation analysis requires a simulation. The usefulness of perturbation analysis is discussed next.

A straightforward approach to apply perturbation analysis to our call center is by means of simulation. It allows one to evaluate and compare the system performance for different parameters of the control method, which are the waiting time factors in this chapter. Next, the best-performing parameter is preferred to control the future evolution of the process. Obviously, to obtain the most accurate results, one should simulate the system as it is expected to behave in the near future. If the future arrival rate is unknown we decided to simulate the most recent history of the process. Besides, we applied stochastic coupling between the different simulations to increase the accuracy of the comparison, with respect to the arrival epochs and the service times.

We can argue that perturbation analysis is not very useful to the way of updating the waiting time factors. We conjecture that our way of the assigning waiting time factors to the queues gives (almost) the highest-possible responsiveness. Note that, in case of stationary Poisson processes, the arrival rates are memoryless such that the future evolution is independent of the past. Hence, the responsiveness gets worse by using perturbation analysis. This was confirmed by our experiments. Furthermore, the sample paths and therefore performance measures do not change continuously as the value of beta changes. Hence, the gradient estimates using perturbation analysis are biased, and may not point in the right direction.

In our opinion, perturbation analysis could be useful for other purposes. In particular, to optimize the value of β. This has the potential benefit to reduce the variances of the waiting times, while meeting the targets concerning the service levels. This is a subject for further research.

6.4 Numerical results

We executed a number of experiments to evaluate the effectiveness and performance of the control policy. The performance is evaluated in different ways, by measuring the objective value from Equation (6.2), calculating the variances and averages of the waiting times, and measuring the service levels. Call centers with two and three skills are considered.

6.4.1 Performance measures

To analyze the quality of the control method a large number of runs were executed, denoted by I, each resulting in a number n_m^i and vectors a_m^i and

η_m^i, indexed by a superscript. The I periods together are considered as one long simulation run. The performance measures are discussed next.

Three skills

To verify if the targets c_m are met we calculate the average abandonment rate over all I runs of each type m

$$\bar{q}^m = I^{-1} \sum_{i=1}^{I} \left(\frac{\sum_{j=1}^{n_m^i} a_m^i(j)}{n_m^i} \right),$$

and we also calculate the average abandonment rate over all types

$$\bar{q} = I^{-1} \sum_{i=1}^{I} \left(\frac{\sum_{m=1}^{M} \sum_{j=1}^{n_m^i} a_m^i(j)}{\sum_{m=1}^{M} n_m^i} \right).$$

The average waiting time is calculated by

$$\bar{\eta}^m = I^{-1} \sum_{i=1}^{I} \left(\frac{\sum_{j=0}^{n_m^i} b_m^i(j)}{(n_m^i - a_m^i e)} \right),$$

with $\bar{\eta}$ denoting the weighted average over both types, and e is a vector with the value 1 at each position. The average waiting time is determined by the average waiting time of the served customers. We remark that there are different ways to define and calculate the average waiting time. A more advanced method is to suppose that tagged customers with an infinitely long patience enter the system randomly in time and to use their waiting times.

The standard deviation of the waiting times is calculated by

$$\tilde{\eta}^m = \sqrt{\frac{\sum_{i=1}^{I}\sum_{j=0}^{n_m^i}(b_m^i(j) - \hat{\eta}^m)^2}{\sum_{i=1}^{I}(n_m^i - a_m^i e) - 1}},$$

with

$$\hat{\eta}^m = I^{-1}\sum_{i=1}^{I}\left(\frac{\sum_{j=0}^{n_m^i} b_m^i(j)}{n_m^i}\right),$$

and we define $\tilde{\eta}$ as the weighted average over both types.

Two skills

Concerning the target, the average ratio of the percentages customers with a bad service between both types is calculated by

$$\bar{\alpha} = I^{-1}\sum_{i=1}^{I}\left(\frac{\sum_{j=1}^{n_1^i} a_1^i(j)}{n_1^i} \Big/ \frac{\sum_{j=1}^{n_2^i} a_2^i(j)}{n_2^i}\right).$$

To measure responsiveness, the performance of each run i is measured by

$$\alpha(a^i) = \frac{Q(a_1^i)}{Q(a_2^i)},$$

with $Q(a_1^i)$ and $Q(a_2^i)$ defined according to (6.1), while considering the arrivals of type 1 and 2, respectively. We measure the standard deviation of $\alpha(a^i)$ over I runs, which we calculate in the usual way by

$$\tilde{\alpha} = \sqrt{\frac{\sum_{i=1}^{I}(\alpha(a^i) - \hat{\alpha})^2}{I-1}}, \text{ with } \hat{\alpha} = \frac{\sum_{i=1}^{I}\alpha(a^i)}{I}.$$

6.4.2 Experiments with two skills

Examples of call centers with two skills are analyzed. The analysis consists of three parts: We consider the simulation of a long period first. In the second part of this section, we analyze the responsiveness of the control method by considering many short simulation periods. In the third part we aim to simulate reality. Hence, we measure the performance in case that the workload fluctuates during a period of one day. The service level is defined as the abandonment percentage. Each section is started by mentioning the method that is analyzed.

Long period

Method 1: In this section we analyze the performance of the control policy in case of constant arrival rates, during a long period, i.e., $I = 1$ and h large. Notice that if the interval h becomes very long and the arrival rates are constant, the objective of c_1/c_2 will surely be reached, if the ratio is feasible. We consider the model with parameters $\lambda_1 = \lambda_2 = 50, \mu_1 = \mu_2 = \bar{\mu}_1 = \bar{\mu}_2 = 1/5, \gamma_1 = \gamma_2 = 1/3, s_1 = s_2 = 0, \bar{s} = 480$, and the target ratio is $c_1/c_2 = 0.8$. Waiting times are expressed in minutes.

The simulation is executed for different values of β between 0 and 1. The results are given in Figure 6.1. It shows that for values of β smaller than 0.7 the objective of $c_1/c_2 = 0.8$ is reached accurately by $\bar{\alpha}$. Between 0.7 and 1.0 the desired ratio cannot be reached, since the ratio $\bar{\alpha}$ increases from 0.8 to 1.0. The figure also shows that the variance $\tilde{\eta}$ decreases when β increases. This is not a surprise because it is well known that for the $M/M/s$ queue the first-come-first-serve policy minimizes the variance of the waiting times, see for example Randolph (1991).

Short feedback periods

Method 1: In this section the updates of the control policy occur without using information of the whole history of the system. We consider I short successive time intervals of length h and updates are based on the information that has been observed only in the current interval. The first interval is $(0, h)$, the second $(h, 2h)$, the third $(2h, 3h)$, ..., and the last $((I-1)h, Ih)$. Thus, the updates in a certain interval are independent of the service levels from previous intervals, because the waiting time factors are updated by

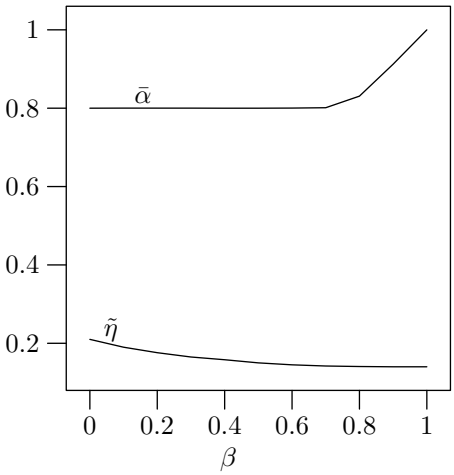


Figure 6.1: Long simulation period for different β

using estimates of the arrivals and the bad services in the current interval only.

By considering short intervals we analyze the responsiveness of the control mechanism. We are interested in how close the ratio between the service levels of the different intervals varies around the objective c_1/c_2. The smaller the average deviation over time, the higher the responsiveness is. We consider a large number $I = 10^4$ of intervals, with a length of $h = 10$ minutes each. The system is not initialized at the beginning of each interval, such that (almost) stationary behavior is measured.

We consider the same parameters as before. The results are presented in Figure 6.2 for different values of β. For each value of β we simulated 10^4 intervals with a duration of 10 minutes. The figure shows that the ratio $\bar{\alpha}$ approaches 0.8 as β goes to 0 and its standard deviations also becomes smaller. The consequence is that the standard deviation of the waiting times increases. We note that the standard deviation of jobs of type 2 is higher than of jobs of type 1. The difference between the standard deviations is highest if $\beta = 0$ and equal to 0 if $\beta = 1$, because of symmetry. Moreover, the total abandonment rate is the same for the different values of β, see Theorem 6.2. The average abandonment rate over both types is around 4.7% and the average waiting times are around 0.14 minutes, which is almost

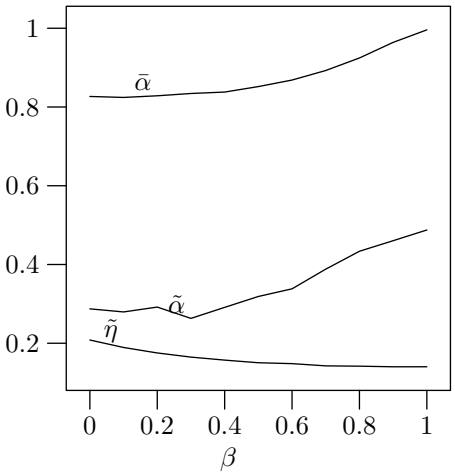


Figure 6.2: Short simulations for different β

independent of β in this example.

To investigate the possible drawbacks of using a dynamic control rule, we compared it to the best static policy that we could find: we chose the highest value of β such that on average c_1/c_2 is reached. This yielded the value $\beta = 0.79$, and $\bar{\alpha} = 0.80$, $\bar{\eta} = 0.140$, $\tilde{\alpha} = 0.569$, $\tilde{\eta} = 0.142$, and 4.58% abandonments. By comparing the static policy against the dynamic policy, we conclude that the average waiting time is a bit higher under the dynamic policy. Under the static policy, the variance in the waiting times is a bit smaller, while the variance of α is much higher. We conclude that the performance of the dynamic rule is relatively good. Moreover, a static policy has the disadvantage that the variance of α is relatively high, even when β is optimized by using a-priori information.

Figure 6.3 shows the performance for different lengths of time intervals. The length is denoted by h and expressed in minutes. All parameters are the same as above, with the exception that $\beta = 0$ is fixed during the experiments. We conclude from the figure that even when using a short history of information, α becomes close to c_1/c_2. This shows that the responsiveness of the online policy is high. Moreover, the values of $\bar{\eta}$ and $\tilde{\eta}$ are similar to the values from Figure 6.2, when taking $\beta = 0$.

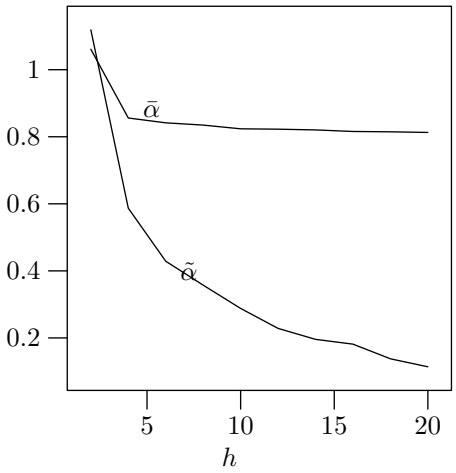


Figure 6.3: Short simulations for different h

Method 2: We consider feedback periods of $h = 10$ minutes, similar to the experiments of Figure 6.2. The performance is evaluated for different values of η, namely $\eta = u^{0.005}$ for $u \in (0, 1)$. The reason for this mapping from u to η is that changes in performance measures become better visible; $\bar{\alpha}$ increases slowly from 0.8 to 1. To increase the responsiveness we impose η to be bounded between 0.1 and 10. Results are presented in Figure 6.4. An important observation is that $\tilde{\alpha}$ is high, compared to the values presented in Figure 6.2. It even exceeds the standard deviation of $u = 1$ significantly, i.e., $w_1 = w_2 = 1$. Hence, we conclude that the reliability of the control policy is low. This indicates a performance difference between the two control policies, which is in favor of Method 1. We remark that the peak of $\tilde{\alpha}$ can be reduced by imposing a stronger bound on η, instead of the range between 0.1 and 10.

Fluctuating workload during one day

Method 1: In this section we simulate the system with a realistic arrival rate pattern during a period of 12 hours. We let the arrival rate of type 1 increase linearly during the first 3 hours and next decrease linearly during the second 3 hours. This pattern repeats during the second 6-hour period. The pattern of type 2 is the opposite of type 1; it increases when the rate of

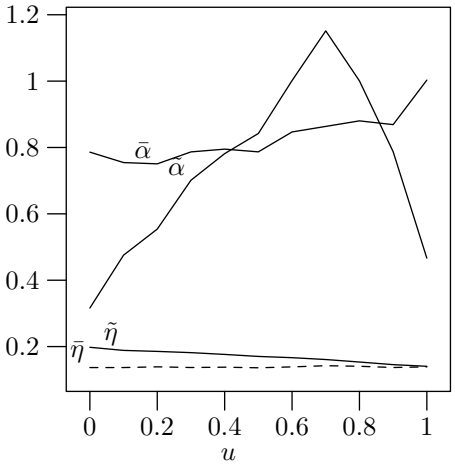


Figure 6.4: Short simulations for different u

type 1 decreases and vice versa, see Figure 6.5. The other parameters are: $\mu_1 = \mu_2 = \bar{\mu}_1 = \bar{\mu}_2 = 0.2, \gamma_1 = \gamma_2 = 1/3, s = 48, c_1/c_2 = 0.8$.

In the experiments we repeated the simulation $I = 250$ times, and started each run with an empty system. The average abandonment rate over both types turned out to be 9.0 percent. Additional results are shown in Figure 6.6. The figure presents the average of α at the end of the day, the standard deviation of α over all runs, and the average waiting time and its standard deviation. We observe that the patterns of the lines look similar to the ones from the previous sections. We also read from the figure that the average waiting time is the same for the different values of β, which we expected due to the choice of parameters, see the results from Section 6.2.3. The same holds for the average abandonment rate.

Method 2: We consider the same situation as we considered for Figure 6.6. We analyze the performance for different values of η, and let $0.1 \leq w_2 \leq 10$. The results are presented in Figure 6.7. In this example, η is given by the relation $\eta = u^{0.0001}$. We note that the lines are not monotonically increasing or decreasing, which we explain by the typical daily pattern of the arrival rates from Figure 6.5. Our conclusion is that the performance is not significantly better than Method 1, but only a bit worse. In particular, $\tilde{\alpha}$ is higher, which is undesirable.

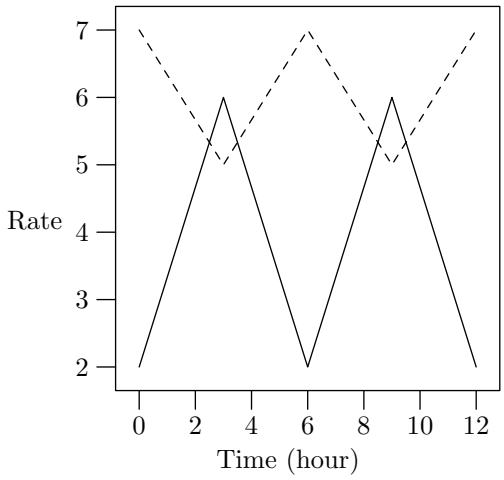


Figure 6.5: Arrival rates

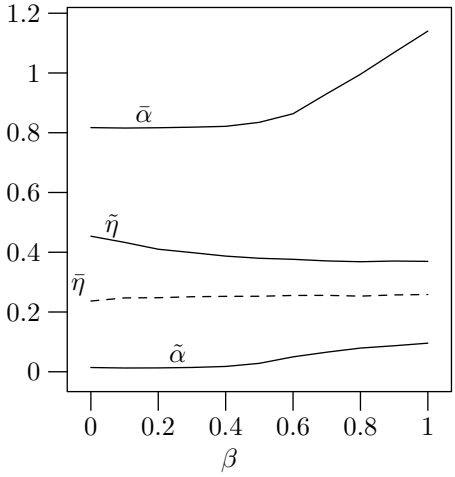


Figure 6.6: Simulations of a day

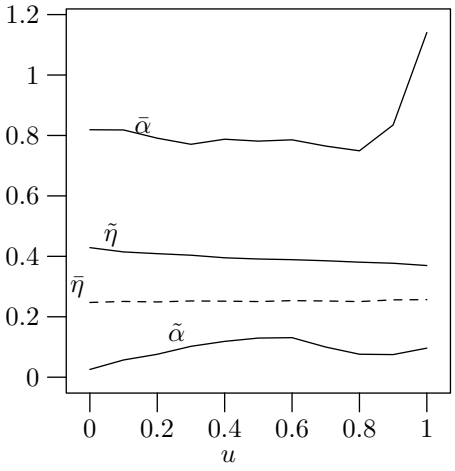


Figure 6.7: Simulations of a day

Method 1: In addition, we evaluate how the ratio α behaves during the day. Therefore, we determine the average of $\bar{\alpha}$ over all I runs, for each hour of the day. The results are presented in Figure 6.8. Each line corresponds to a different value of β. The figure shows that for $\beta = 1$ the ratio converges to 1.15, which is far from 0.8. For smaller values of β the ratio is reached closer, and eventually, for $\beta = 0$, the ratio reaches almost 0.8 on average at the end of the day.

From all previous experiments, we conclude that the control mechanism performs well in realistic scenarios. Values around 0.5 of β yields a good performance with regard to the main objective and the variances.

6.4.3 Experiments with three skills

In this section, we evaluate if Method 1 is also useful in an environment with multiple agent groups, different service rates, and more than two job types. Hence, in the next two sections we consider examples with different parameters and different definitions of the service level: the abandonment percentage and the percentage of customers that are not served within twenty seconds, respectively.

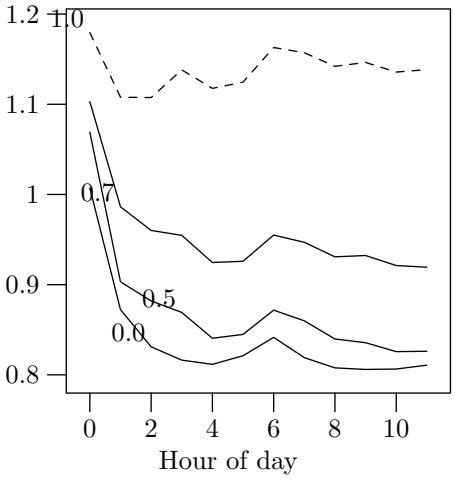


Figure 6.8: Ratio during the day for different values of β

Abandonments

We consider a three-skill call center, see Figure 5.1, with the following parameters: The arrival rates λ_1 and λ_2 are defined according to Figure 6.5, and $\lambda_3 = 4$. Specialists work slightly faster than generalists and the service rates differ per job type: $\mu_1 = 4.5, \mu_2 = 5, \mu_3 = 5.5, \bar{\mu}_1 = 4, \bar{\mu}_2 = 4.5$, and $\bar{\mu}_3 = 5$. To obtain sufficient flexibility of agents we staff more generalists than specialists, $s_1 = 10, s_2 = 5, s_3 = 0$, and $\bar{s} = 53$. The abandonment rates are $\gamma_1 = 1/4, \gamma_2 = 1/3$, and $\gamma_3 = 1/2$. With regard to the updates of the control policy, we set $\beta = 0$.

Table 6.1 presents the results of the experiments. The numbers in the table are averages over $I = 250$ intervals, with a length of $h = 720$ each. The left column shows the parameters of the objective function c_m, with $m \in \mathcal{M}$. To get insight in the performance we evaluated different values of β, presented in the second column. The other columns show the outcomes of the performance measures. We conclude from the results that the average abandonment rate $\bar{q}$ is quite the same but not completely for different values of β. This is related to the different service times and the presence of groups with specialized agents. In contrast, the average $\bar{\eta}$ and the variability $\tilde{\eta}$ of the waiting time differ significantly for the different values of the waiting time factors. The difference in the average waiting time for the different values

(c_1, c_2, c_3)	β	$\bar{\eta}$	$\tilde{\eta}$	$\delta(\bar{q}^1, \bar{q}^2, \bar{q}^3)$	$\bar{q}$
$(1, 3, 6)$	1	0.089	0.201	$(1.9, 3.0, 4.7)$	0.036
$(1, 3, 6)$	0.75	0.089	0.199	$(1.8, 3.0, 5.2)$	0.037
$(1, 3, 6)$	0.5	0.085	0.198	$(1.6, 3.0, 5.7)$	0.037
$(1, 3, 6)$	0.25	0.084	0.202	$(1.3, 3.0, 5.9)$	0.037
$(1, 3, 6)$	0	0.081	0.230	$(1.1, 3.0, 6.0)$	0.037
$(1, 1, 1)$	1	0.091	0.202	$(0.6, 1.0, 1.6)$	0.036
$(1, 1, 1)$	0.5	0.093	0.207	$(0.8, 1.0, 1.3)$	0.036
$(1, 1, 1)$	0	0.096	0.246	$(1.0, 1.0, 1.1)$	0.036
$(2, 3, 5)$	1	0.091	0.201	$(1.9, 3.0, 4.7)$	0.036
$(2, 3, 5)$	0.5	0.090	0.206	$(2.0, 3.0, 5.0)$	0.036
$(2, 3, 5)$	0	0.085	0.241	$(2.0, 3.0, 5.0)$	0.036
$(3, 2, 1)$	0	0.099	0.242	$(3.3, 2.0, 2.0)$	0.036

Table 6.1: Results in case of three skills

of β is explained as follows. The waiting time factors determine the service priorities of the jobs. If waiting time factors change, this has an impact on the queue lengths and changes the average amount of work in each queue. Due to the different service times among the job types, the average number of jobs in both queues changes. Consequently, it changes the abandonment rate of each job type and the average waiting times as well. For example, if jobs with short waiting times are served instead of jobs with long waiting times, the average waiting time decreases. The change in the variability of the waiting times is more difficult to explain.

We conclude from the experiments that the objective values c_m are in most cases accurately achieved. In the other cases, either the type of routing policies is too restrictive, or there is a mismatch between the service capacity and the workload of certain job types. To obtain better results, non-work-conserving policies would be required, for example by means of thresholds. Another way is the rescheduling of agents from one group to another, such that the amount of workforce is in balance with the offered workload and the desired service levels. This can yield, to a certain extent, a similar improvement as introducing a threshold. While thresholds are theoretically maybe nearly optimal, rescheduling agents may be easier to implement and to analyze. This is a subject for future research.

Long waiters

We consider the same parameters as in the previous section, except that $\bar{s} = 50$. Table 6.2 presents the results of the experiments. It contains the same type of data as Table 6.1, but with a few differences: The service-level measure is the percentage of customers that wait longer than twenty seconds, compared to the abandonment rate in Table 6.1. That is why, in addition, the abandonment rates are also given, see the most-right column. The time units of all rate parameters are minutes.

Comparison

In this section we compare the service-level definitions from the previous sections. Consider the values from Tables 6.1 and 6.2. We conclude that sometimes the objective is met more accurately with the definition on the waiting time, than with the service level being defined as the abandonment rate. For example, the difference is visible when studying the results in the lowest row of both tables. An explanation is given next. The abandonment rate of type 3 is relatively high, since the group of dedicated agents with skill 3 is empty. Consequently, the objective $(3, 2, 1)$ is not achieved in Table 6.1. In contrast, the abandonments have a positive impact on the results of Table 6.2. The abandonments reduce the waiting time of the customers in the queue, which decreases the number of customers that wait longer than the AWT, such that the objective can be met. Hence, in this example the objective is achieved more easily when using the service-level definition on the waiting-time distribution. However, this is not always the case; abandonments can also have a negative impact on the results from the previous section. This is visible, for example, in the sixth row of both tables. It shows that the objective $(1, 3, 6)$ is more difficult to achieve when using the waiting-time definition. This can be explained similarly.

6.5 Concluding remarks

This chapter introduced two methods to optimize the service levels based on real-time information. Although both methods perform well, our preference goes to one of them, because of its simplicity. Furthermore, this method requires no specific information about the system.

An interesting subject for further research is to optimize the total number

(c_1, c_2, c_3)	β	$\bar{\eta}$	$\tilde{\eta}$	$\delta(\bar{q}^1, \bar{q}^2, \bar{q}^3)$	$\bar{q}$	Perc. Aband.
$(1, 3, 6)$	1	0.142	0.257	$(2.5, 3.0, 3.1)$	0.186	(3.4,5.4,8.5)
$(1, 3, 6)$	0.75	0.137	0.252	$(2.3, 3.0, 3.3)$	0.180	(3.0,5.2,9.1)
$(1, 3, 6)$	0.5	0.133	0.248	$(2.0, 3.0, 3.7)$	0.173	(2.6,5.0,10.0)
$(1, 3, 6)$	0.25	0.124	0.243	$(1.7, 3.0, 4.4)$	0.149	(2.1,4.3,11.3)
$(1, 3, 6)$	0	0.106	0.256	$(1.2, 3.0, 5.8)$	0.107	(1.4,3.5,13.7)
$(1, 1, 1)$	1	0.141	0.256	$(0.8, 1.0, 1.0)$	0.186	(3.4,5.5,8.6)
$(1, 1, 1)$	0.5	0.144	0.264	$(0.9, 1.0, 1.0)$	0.185	(3.9,5.3,8.2)
$(1, 1, 1)$	0	0.134	0.313	$(1.0, 1.0, 1.0)$	0.143	(3.9,5.0,8.5)
$(2, 3, 5)$	1	0.142	0.201	$(2.5, 3.0, 3.1)$	0.185	(3.4,5.4,8.5)
$(2, 3, 5)$	0.5	0.132	0.248	$(2.3, 3.0, 3.8)$	0.171	(2.7,4.8,10.1)
$(2, 3, 5)$	0	0.115	0.279	$(2.0, 3.0, 5.0)$	0.147	(2.0,3.9,12.6)
$(3, 2, 1)$	0	0.157	0.313	$(2.8, 2.0, 1.2)$	0.162	(7.0,5.1,5.2)

Table 6.2: Results in case of three skills

of agents and the skills that are used by the agents, by adjusting the sizes of the agent groups. We expect that the control policy can also be applied successfully to systems with additional groups of cross-trained agents. Then it is natural to assign the same waiting time factor for each call type to the different groups.

Chapter 7

Approximate Performance Evaluation

The subject of this chapter is to measure the performance of service in a multi-skill call center based on system parameters. This can be useful for optimization purposes.

7.1 Introduction

The call center is modeled as a network with multiple arrival streams. Each arrival stream is a Poisson process and contains its own type of jobs. The network consists of groups with multiple servers, and agents of the same group have the same set of skills. The routing of jobs to servers or agents occurs according to a fixed overflow routing policy. According to this type of policy a call is assigned to the server with the highest priority. These priorities specify the policy uniquely in addition with: (1) jobs are rejected when all servers with the required skill are busy and (2) jobs leave the network immediately after service completion. Inter-arrival and service times are exponentially distributed with different parameters for different job types. The main subject of this chapter is the approximation of the blocking probabilities.

An example of applying the blocking model to call centers is optimizing the performance for a fixed number of agents by dividing the agents among the groups. It enables call centers to take the skills of the available employees into account in the optimization of service levels. However, this is only useful if the optimal solution of a blocking model will also perform well if queueing

is allowed, which is therefore analyzed in Chapter 8. An example of such an optimization algorithm is also given in Koole, Pot, and Talim (2003).

Possible applications, other than call centers, can be found in the area of tele-communication and computer networks. A possible application in telecommunication are joint ventures between mobile phone service providers to share regions of their bandwidth. During intervals that a provider has no bandwidth, additional resources are available in these shared regions. In the area of computer networks we could think of webhosting. Consider a group of servers that handle the service request of some internet pages. Each server is dedicated to serve the requests of a subset of these pages. For economic reasons it may be desired to minimize the total number of servers that treat a certain page. On the other hand, due to service level requirements, it may be attractive that the network also contains servers that can handle multiple pages. This could be modeled similarly.

Our approach is to consider each group separately, as a $(\sum G/M)/s/s$. The notation $(\sum G/M)/s/s$ denotes that there are multiple arrival streams with different inter-arrival and service time distributions. A difficulty is that an exact analysis of the overflow process of the $(\sum G/M)/s/s$ system is extremely complicated. An accurate description of this process is important because it determines the arrival processes of the next groups. Of course, it is desired that also the next groups are described as accurately as possible as a $(\sum G/M)/s/s$ system.

Since not much is known about the $(\sum G/M)/s/s$ system we considered approximation algorithms. A first step is made by Koole and Talim (2000), in which the overflow processes are approximated by Poisson processes. No distinction is made between the different types; weighted average service times per group are taken. This method will be referred to as the Exponential Decomposition (ED) method in this chapter. From the research in Koole, Pot, and Talim (2003) we concluded that a better approximation method is desired. This is accomplished by approximating each group by a $(\sum H_2/M)/s/s$ model (with H_2 denoting second-order hyperexponential inter-arrival times), and we will refer to this method as HyperExponential Decomposition. An advantage of independent hyperexponential arrival streams is that the first moment of the overflow process can be calculated exactly. We also succeeded in finding a good approximation for obtaining higher moments. The approximation method is treated in this chapter. Furthermore, a comparison with regard to alternative methods, such as the ED method, is provided.

Several other methods exist that also implicitly take higher moments of the overflow processes into account. These are discussed next.

Equivalent Random Method

The first method we introduce is a well-known method from the literature, called the Equivalent Random Method (ERM), see Cooper (1981) pages 165–171. This method serves for approximating blocking probabilities of overflow networks. It is based on a formula for the peakedness of the overflow process of an $M/M/s/s$, developed by Wilkinson (1956) and Bretschneider (1956) and also given in Kosten (1973). Wilkinson (1956) developed approximation tables and he gave examples of the ERM. Jagerman (1984) presents an iterative method and generalizes Kosten's formula for systems with general service time distributions. Rapp (1964) developed an approximation formula.

While in the literature the ERM is often described for networks with two layers, it can be generalized to multi-layer networks, see for instance Tabordon (2002). A limitation is that the ERM can only be applied to networks with a tree structure (splitting of overflow traffic is not supported). Moreover, the ERM does not allow different service times for different customer classes. However, more literature is available about this subject, which can be found Schehrer (1997).

To overcome the limitations, the ERM was generalized to more general overflow routings in Chapter 4 of Tabordon (2002). The generalization was inspired by procedures from other methods, which are presented next.

We note that this extended ERM is not included in our numerical comparison, because the performance is dominated by most of the other methods. For further details, see Tabordon (2002).

Hayward-Fredericks Method

The Hayward-Fredericks method (HF) is an extension of the ERM. Hence, we refer the reader to the same literature. This method is also called the Equivalent Congestion Method, see for example Sanders, Haemers, and Wilcke (1983). For server groups with Poisson input the procedure of the HF is similar to the ERM. However, if groups in the system receive overflow streams from other server groups as input, the approach is different. While in the ERM multiple groups are replaced by one 'equivalent' group, the HF consid-

ers each server group separately. This became possible due to Hayward and Fredericks (1980). We note that a reference to the work of Hayward cannot be given, because it has never been published in his name. (Therefore, we mention that it is given as a reference in several books and papers, see for example Wolff (1989) pages 354–355.) The contribution of Hayward is that he found a good approximation for the blocking probability of a multi-server group with peaked input. As a result, the blocking probability of two-layer networks can be approximated accurately. Fredericks found an approximation for the peakedness factor of the overflow stream of systems with peaked input, which made it possible to analyze networks with more than two layers. These results were applied to call centers in Chevalier and Tabordon (2003). By using their results, we will compare our method to the HF method.

Interrupted-Poisson-Process Method

The name Interrupted-Poisson-Process method (IPP) denotes the type of process by which the overflow streams are approximated. The overflow of a group with Poisson input was analyzed by Kuczura (1973). The method is extended in Chevalier and Van Muylder (2001) to the overflow process of a group with an IPP as input, in which it also is evaluated and applied to call centers. Their contribution also consists of an efficient method to approximate the superposition of n IPP processes and a way to dispatch the overflow process among the next groups. For further details about these methods and a comparison we refer the reader to Tabordon (2002) and Van Muylder (2001).

HyperExponential-Decomposition Method

The approximation method presented in this chapter was developed independently from the work of Chevalier, Van Muylder and Tabordon. It uses different methods for calculating both blocking probabilities and higher moments of the overflow streams. Besides, the dispatching of an overflow stream to several destinations is handled in an alternative way.

This chapter is structured as follows. In Section 7.2 we introduce the model and notation. An exact solution approach is only feasible when the total number of states is moderate, e.g., one thousand states. For larger state spaces we suggest to use the heuristic algorithm as is presented in

this chapter. The main idea behind the algorithm is a reduction of the state space. This is realized by decomposition. The problem size is reduced by considering each server group separately. The arrival processes to each group are approximated by multiple hyperexponential arrival streams of order 2, hence it is denoted as the HyperExponential Decomposition (HED) method. Section 7.3 presents relevant results about the overflow process of an $M/M/s/s$ system. This basic overflow process plays an important role in the algorithm that is presented in this chapter. Section 7.4 describes the decomposition algorithm; an exact analysis is applied to each group separately. Section 7.5 shows the benefit of the HED method in comparison to simulation. In Section 7.6 we refer to Appendix A, which contains a number of numerical examples. Further, we show that the HED method is more accurate than the ED method, and the benefit is high in certain cases. This is achieved by means of several examples. We also consider the other methods: the HF method, the ERM, and the IPP method. Finally, Section 7.7 gives the conclusions, possible extensions and other ideas.

7.2 Model description

An example of the model is depicted in Figure 7.1, which is the graphical representation of the considered blocking system. This figure completely

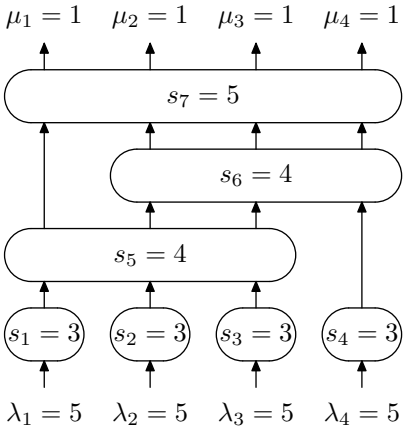


Figure 7.1: Illustration of the call center model

defines the system. Each arrow at the bottom represents the arrival process

of a certain job type (or class).

The following notation is used:

$\mathcal{M} \subseteq \mathbb{N}$ is the index set of job classes. All jobs from a certain class are generated by the same 'source'. Therefore, $\mathcal{M}$ will also be referred to as the set of sources.

$\mathcal{G} \subseteq \mathbb{N}$ is the index set of server groups.

S_g is the subset of job classes that server group g can handle, with $g \in \mathcal{G}$ and $S_g \subseteq \mathcal{M}$.

μ_m specifies the service rate of class m, with $m \in \mathcal{M}$.

λ_m is the rate of the Poisson process by which jobs from class m arrive.

s_g specifies the size of server group g, with $g \in \mathcal{G}$.

$g(m, j)$ denotes the j-th server group in the overflow routing policy for source m, with $m \in \mathcal{M}$ and $g(m, j) \in \mathcal{G}$.

Process m is a Poisson process with rate λ_m, and m denotes the job type. The service time distributions are assumed to be exponential. The service rate of job type m is μ_m. In this chapter we assume that the service rates do not depend on the server or server group. However, this assumption is not essential. The HyperExponential-Decomposition method can also be applied to a system with different service rates for different groups. In addition, each circle represents a server group. The number s_g of servers in group g is given in the middle of each circle. The incoming Poisson streams, as well as the overflow streams, are depicted by arrows. Job assignment occurs according to a fixed overflow route. This means that there exists an ordering of groups for each job type such that a job is assigned to the first group with an available server. Jobs are rejected when all servers along the overflow route are busy. Served jobs leave the system immediately.

The blocking system in Figure 7.1 could be described as:

$$\mathcal{M} = \{1, 2, 3, 4\}, \qquad \mathcal{G} = \{1, 2, 3, 4, 5, 6, 7\},$$
$$S_g = \{g\} \quad \forall g \in \mathcal{M}, \quad S_5 = \{1, 2, 3\}, \quad S_6 = \{2, 3, 4\}, \quad S_7 = \{1, 2, 3, 4\},$$
$$\mu_m = 1 \text{ and } \lambda_m = 5, \quad \forall m \in \mathcal{M},$$

$s_1 = s_2 = s_3 = s_4 = 3, \quad s_5 = s_6 = s_7 = 4,$

$g(1,\cdot) = (1,5,7), \quad g(2,\cdot) = (2,5,6,7), \quad g(3,\cdot) = (3,5,6,7), \quad g(4,\cdot) = (4,6,7).$

This model formulation is quite different from the one considered by Van Muylder (2001) and Tabordon (2002). In this chapter the service rates only depend on the job class, whereas in their work they are group-dependent and job-independent. However, including group-dependent service rates creates no fundamental problems in our method. Furthermore, we consider overflow policies that are deterministic, instead of randomized. Deterministic overflow policies route each job through the network in a deterministic manner, while randomized policies choose the next group randomly with some fixed probabilities. In Tabordon (2002) these probabilities are optimized such that the workload is balanced among the different agent groups. This load balancing is essential for their approach. On the other hand, we also consider unbalanced systems, which often occur in reality. Nevertheless, including randomization creates no fundamental problems in our method.

Model limitation

Concerning the approximation algorithm that is discussed in Section 7.4 it should hold that server groups can be numbered in such a way that

$$g(m,i) < g(m,j) \Leftrightarrow i < j, \qquad \forall m \in \mathcal{M}, \forall i,j \in \mathbb{N}.$$

This means that it must be possible to arrange the groups in Figure 7.1 in such a way that there are only upward-pointing arrows.

7.3 Fitting the overflow process of the $M/M/s/s$ model

When the inter-arrival times of jobs to a group with identical servers are independent, it is possible to calculate the first moments of the overflow stream analytically. This is relevant to server groups in the first layer of the overflow routing network. In addition, we know that the independence of the inter-arrival times also holds if the arrival stream of a group is a renewal process originating from exactly one first-layer group. In general, if a group

receives calls that originate from several groups, the independence of the inter-arrival times does not hold anymore. In other words, the superposition of multiple arrival processes is no longer a renewal process. Since we are interested in an algorithm that allows this type of routing, an exact analysis is very difficult. Next, we elaborate on the calculation of the first moments of the $M/M/s/s$ queue. The results are important for the algorithm that is introduced in Section 7.4.

The real overflow process of the $M/M/s/s$ system is a renewal process with a hyperexponential distribution of order $s+1$ and density function, see Riordan (1961), page 39,

$$f(t) = \sum_{i=1}^{s+1} p_i \gamma_i e^{-\gamma_i t}. \tag{7.1}$$

This section deals with fitting this hyperexponential distribution to a lower-order hyperexponential distribution of order $\bar{s}, \bar{s} \leq s$. This reduces the size of the state space to such an extent that the model becomes tractable for blocking systems of realistic sizes.

The method is discussed from a numerical point of view, because analytical expressions for the exact distribution become too large and cannot be determined efficiently. In this calculation we distinguish three steps: (I) determining the coefficients of the real H_{s+1} distribution (given the arrival rate, service rate and the number of servers), (II) determining the first $2\bar{s}-1$ moments and, finally, (III) determining the coefficients of the fitted $H_{\bar{s}}$ distribution. As an alternative, steps (I) and (II) could be replaced by estimating the moments with a simulation table of an $M/M/s/s$ system. This is explained later in this section. First, the three steps are discussed:

(I) We refer the reader to Riordan (1961), pages 36–39. On these pages the relevant equations are given. The computation involves a partial-fraction expansion of a division of two Poisson-Charlier polynomials. This can be solved numerically with standard methods, e.g. the Heaviside cover-up method, which includes a method for finding roots. Mathematical software packages like Maple or Matlab are also capable to handle these. In our implementation we obtained accurate solutions with groups of up to ten servers.

(II) The n-th moment of a random variable X with the density func-

tion of Equation (7.1) is determined by

$$\mathbb{E}\bar{X}^n = \sum_{i=0}^{2\bar{s}-1} n! \frac{p_i}{\gamma_i^n}.$$

This is obtained by using the Laplace transform or moment generating function.

(III) The coefficients of the fitted $H_{\bar{s}}$ are generally hard to calculate, analytically as well as numerically. For H_2 we get, see Tijms (1986b), page 360

$$\gamma_{1,2} = \frac{1}{2}\left\{a_1 \pm \sqrt{a_1^2 - 4a_2}\right\}, \quad p_1 = \frac{\gamma_1(1-\gamma_2 m_1)}{\gamma_1 - \gamma_2}, \quad p_2 = 1 - p_1, \tag{7.2}$$

with

$$a_2 \equiv \frac{6m_1 - 3m_2/m_1}{3m_2^2/2m_1 - m_3}, \quad a_1 \equiv \frac{1}{m_1} + \frac{m_2}{2m_1}a_2, \text{ and } m_n \equiv \mathbb{E}\bar{X}^n.$$

This holds when $m_1 m_3 \geq \frac{3}{2}m_2^2$. There are no such explicit expressions for $\bar{s} > 2$.

Simulation is an alternative for the steps (I) and (II) of the analytical method. This seems to be unattractive, because a drawback of simulation (and even of both numerical methods) are the long computation times. Therefore we suggest doing these calculations in advance.

We calculated the moments for different parameter combinations, a finite number of pairs (s, a), with a the workload and s the number of servers. We let s vary between 0 and 30 and λ from 0 to 50 with steps of 1/8 and fixed $\mu = 1$. The moments for other values of μ are found by rescaling. The outcomes are stored on a hard disk and are loaded before the algorithm is executed. During the algorithm interpolation is applied to find approximations for real values of a.

There exists a different approach to obtain an exact expression for the first $\bar{s}$ moments of the overflow process of an $M/M/s/s$ system. It is based on Takács (1962) and an essential part of the IPP method. It is applied by Van Muylder (2001), and referred to and discussed by Tabordon (2002). It is summarized next because it is also relevant to the algorithm that is

presented. The idea behind the approximation method is to compare two systems, denoted by A and B. System A is a multi-server group with Poisson input, and blocked calls are served by an infinite-server group. System B is an infinite-server group and its inter-arrival time distribution is hyper-exponential. The arrival time distribution in system B is approximated by using the first $\bar{s}$ moments of the overflow distribution in system A. This is achieved by equating the first three moments of the number of jobs in the infinite-server group. In Van Muylder (2001) it is assumed that the arrival process of system A is an IPP. In this chapter we consider this approach to determine the overflow of a group with Poisson input. Since a Poisson process is a special case of an IPP we could use their results.

In our software implementation we have chosen to determine the moments with a simulation table for group sizes larger than ten servers. For group sizes smaller than ten, the exact method for the computation of the first three moments was used (based on Takács). Furthermore, the simulation table was filled in advance, and only once. The reason for using these tables is shorter computation times. We remark that the outcomes of the three different methods are all very accurate.

7.4 HyperExponential-Decomposition algorithm

In this section we give a description of the algorithm. In the algorithm the overflow streams are approximated by hyperexponential distributions. This is depicted in Figure 7.2 for an arbitrarily chosen scenario.

First we consider the most basic case of a call center with two layers; layer 1 consists of specialists and layer 2 of generalists. The algorithm processes the server groups separately and successively, starting with the groups in the first layer. Each group of specialists is modeled as an $M/M/s/s$ system. This is an exact representation within the model. We are interested in the overflow process of each group. These are approximated by second-order hyperexponential distributions. The approximation requires the first three moments of the overflow process. These moments determine the coefficients of the hyperexponential distribution, see Equation (7.2) in Section 7.3. The hyperexponential distributions are taken as input to the group of generalists in layer 2. Next, the group of generalists is considered separately as a system with hyperexponential arrival streams, which can be modeled as a birth-death process. The blocking probability is determined by solving the balance

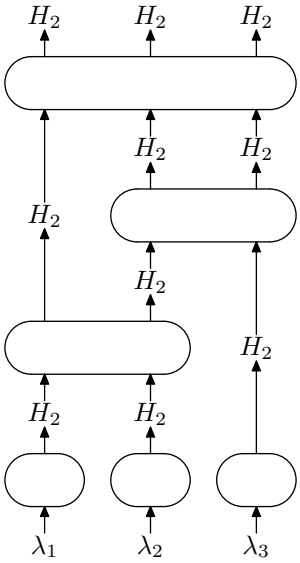


Figure 7.2: Illustration of the decomposition

equations, see Section 7.4.3 and Appendix B.

The algorithm that applies to the general model from Section 7.2 is more elaborate. With regard to the previous example two problems can arise and need to be solved. The first one concerns the splitting of overflow streams. The meaning of splitting is explained next, also by means of an example. Consider a call center with the generalists in the first layer and the specialists in the second layer. The essential difference with the previous example is that the overflow process of the generalists is directed to multiple groups. Since initially one overflow process is determined, this process must be split to the different groups in the next layer. This type of construction is supported in the general algorithm, see Section 7.4.5.

Another problem arises when we consider a call center with three layers. The question is what the overflow process from the groups in layer 2 looks like. In the description above only the calculation of the blocking probability of these groups is treated. However, additional information is required to compute the parameters of the hyperexponential distributions that characterize the processes to the groups in layer 3. More specifically, the first three moments are required. Therefore, an equivalent random process is

taken, derived from the $M/M/s/s$ system. This will also be treated in the remainder of this section, see Section 7.4.4.

The pseudo-code of the HyperExponential-Decomposition algorithm is presented next.

```
HEURISTIC ALGORITHM()
 1   determine the level of each group, define L as the highest level
 2  for  level 0 up to L
 3      do for  each group in current level
 4              do  - calculate the weighted average service rate, per
 5                    arrival stream
 6                  - calculate analytically the blocking probability,
 7                    assuming H2 arrival processes
 8                  - determine the second and third moment of the
 9                    overflow process
10                  - approximate and dispatch the fitted overflow process
11                    to the next groups
```

The steps at lines 1–10 are discussed in the next sections. The number in the brackets refers to the line that will be discussed.

7.4.1 The level of each group (line 1)

The level of each group is derived from the routing policy and calculated by an iterative method.

7.4.2 Weighted average service rate (line 4)

Consider some arbitrary server group g, with skill set S_g. If jobs from different classes arrive from the same preceding group, their arrival processes are no longer independent. An exact analysis of those arrivals will be hard. However, we do not wish to exclude these arrival streams, since they are common in real-life overflow systems. Therefore we use an approximation in which this type of arrival stream is modeled as one hyperexponential renewal process with one exponential service time distribution. Here we use the weighted average service rate of the jobs originating from the same preceding group. The weighted average service rate of the overflow stream

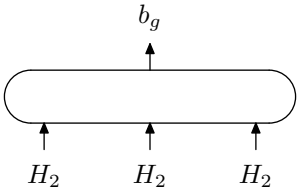


Figure 7.3: Blocking probability for group g

from group i to group g, denoted as $\bar{\mu}_{ig}$, is then determined by

$$\bar{\mu}_{ig} = \sum_{m \in S_g \cap S_i} \lambda_{igm} \left(\sum_{m \in S_g \cap S_i} \frac{\lambda_{igm}}{\mu_m} \right)^{-1}, \tag{7.3}$$

with λ_{igm} the overflow rate of jobs of type m from group i to group g. If the overflow of jobs from group i to group g of type m is not possible according to the routing policy, then the rate λ_{igm} is 0. In case an inflow stream of group g is not an overflow process, we replace λ_{igm} by λ_m.

7.4.3 Overflow rate (line 6)

Let us consider server group g, of size s_g. This section describes the calculation of the blocking probabilities and the overflow rate of this group, as illustrated in Figure 7.3. In the algorithm the inter-arrival times of the overflow streams to group g are fitted by hyperexponential distributions of order 2. If we assume the arrival streams originating from different groups to be mutually independent, we can model the sub-system of group g as a birth-death process, in which each arrival stream has two possible states. As a result, the number of states of the system that is relevant for group g becomes $m(g)2^{n(g)}$ with $n(g)$ the number of immediately preceding groups from which the arrival stream originates and $m(g)$ the number of states of group g. A more extensive explanation is given in Appendix B.

Inaccuracy is caused by the fact that we determine a weighted average service rate of the jobs that originate from the same group, as described in Section 7.4.2. This averaging of the service times is not fundamental to the method but reasonable. It is included for the sake of reducing the state space of decomposed group g and can therefore be left out when the number of servers in group g is relatively low.

To obtain the blocking probabilities, we solved the balance equations, see Appendix B. The overflow rates follow directly from the equilibrium state probabilities. P_{jig} denotes the total equilibrium probability of the source generator from i to g being in state j, while at the same time all servers in group g are busy. Now, λ_{ig}, the total overflow rate at group g of jobs originating from group i can be expressed as

$$\lambda_{ig} = \sum_{j=1}^{2} P_{jig}\gamma_{jig},$$

with γ_{jig} the rate of the generator from group i to group g when this generator is in state j. In case of combined input streams, the λ_{igm} as defined in Section 7.4.2 can be calculated by assuming that the fractions of different types in combined streams remain equal. The overall blocking probability up to group g becomes

$$b_g = \frac{\sum_{m \in S_g} \lambda_{\cdot gm}}{\sum_{m \in S_g} \lambda_m}$$

and will be used in Section 7.4.4. The dot in the equation denotes a summation of rates over all previous groups g. However, there can be only one positive rate for each combination of g and m by our definition of the routing policy. The rates of type m associated with all the other previous groups of group g are zero.

7.4.4 Second and third moment of the overflow (line 8)

In this section a method is described for finding an approximation of the second and third moments of the inter-overflow times of group g. We assume that the blocking probability of group g has been determined according to the previous section. This probability will be denoted by b_g.

The second and third moments are determined next, by interpolation. See Figure 7.4 for an illustration. The general idea is to compare the complete overflow process up to group g to an $M/M/s/s$ system with the same blocking probability, under the assumption that the burstiness is then comparable as well. Firstly, by using $B(s, a)$, the Erlang loss formula, we deter-

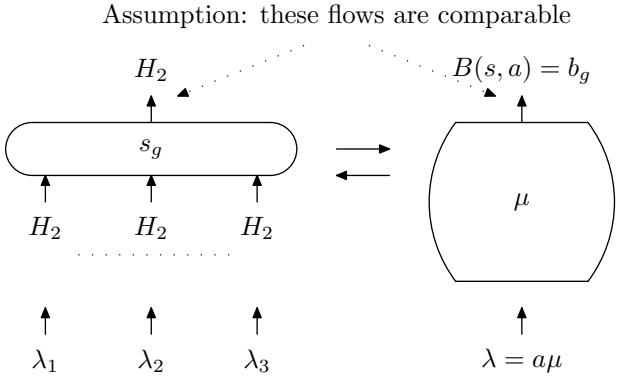


Figure 7.4: Second and third moment estimation by comparison

mine a lower and upper bound for s as follows

$$s_L := \max\{s \in \mathbb{N} : B(s,a) \geq b_g\},$$
$$s_U := \min\{s \in \mathbb{N} : B(s,a) \leq b_g\},$$

with

$$a := \sum_{m \in S_g} \frac{\lambda_m}{\mu_m}.$$

Secondly, for both s_L and s_U the first three moments of the overflow processes are determined according to the formula from the $M/M/s/s$ system, as described in Section 7.3. Finally, linear interpolation between s_L and s_U is applied to obtain the final values for the moments. This is done in such a way that the first moment matches with b_g.

We conclude from our experiments that the accuracy of the higher moments is comparable to the accuracy of b_g.

7.4.5 Overflow to the next groups by dispatching (line 10)

For each group the first three moments of the inter-overflow times are calculated in line 8. These moments are used to uniquely fit a hyperexponential distribution of order two, of which the density function is given in Equation (7.1). In general, this overflow stream contains different types of jobs. We need to keep in mind that the overflow processes of these different job types are not independent. Therefore, dispatching of overflow streams only

takes place between different destinations, not between job types. So, if the overflow stream is directed to only one higher level group, there is no need for dispatching, even if there are different job classes in the stream. On the other hand, if the overflow stream is directed to l different higher level groups, the stream needs to be split into l substreams. Each of these substreams may contain several job classes, see Figure 7.5 for the case $l = 3$. In general, an

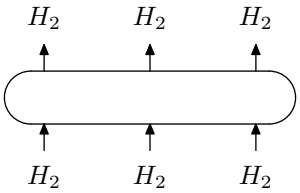


Figure 7.5: Dispatching

exact analysis of this dispatching problem is extremely complicated. However, it turns out that excellent results can be obtained by assuming that the dispatched substreams are renewal processes with second-order hyperexponential inter-event times. For the parameters of these hyperexponential distributions we use the same p_i's as found for the hyperexponential fit of the total overflow stream. For the γ_i's of the substream from group g to k, the γ_i's of the total overflow stream are multiplied by p_{gk}, the fraction of the total overflow stream from group g that is directed to group k. This fraction is determined by the following summation:

$$p_{gk} = \frac{\sum_{m \in S_g \cap S_k} \lambda_{gkm}}{\sum_{m \in S_g} \lambda_{g \cdot m}},$$

with λ_{gkm} the overflow rate of jobs of type m from group g to group k and the dot having the same meaning as before in Section 7.4.3. Here, we define 'rate' as the inverse of the expectation of the concerning inter-event times.

If the inflow of group g consists of a separate hyperexponential stream per skill type m, then λ_{gkm} follows directly from Appendix B. If an inflow stream of group g contains of multiple job classes, then only the total overflow rate per input stream follows from the exact analysis. Therefore, we assume that the fractions remain equal in the inflow and outflow. For example, consider

the case that S_k, the skill set of group k, cannot be composed by taking the union of job classes from inflow streams of group g. Then the λ_{gkm} is obtained by taking the overflow rate of the input stream that contains type m, and multiplying this rate by the probability that a job in the inflow stream is from type m.

There is also literature available about splitting streams of traffic, see for example De Boer (1985) and Rajaratnam and Takawira (1997). The methods are not applicable to our model because their routing policies are very limited and multiple skills and different service rates are not allowed.

7.5 Comparison to simulation

In this chapter simulation is used to validate the accuracy of the approximation methods. The resulting values are considered to be the true value since simulation is the only way to carry out the calculations without unproven assumptions.

In this section we focus on the computation times of the HED method and simulation, and especially on the differences between both methods. The accuracy will be compared in Section 7.6.

The comparison is made by analyzing the call center from Figure 7.6. All computations were executed on an Asus laptop with an Intel Pentium III 1066 Mhz processor. Execution of the HED method took 1.7 millisecond. The estimates of the blocking probability by simulation is displayed against time in Figure 7.7. In total, each run consists of 1.5×10^6 simulated events. We observed that the blocking probability converges slowly. Even after one second the two most extreme probabilities of ten sample paths differ by 0.005. The relative difference is about 4%. In comparison to the HED method, which yields a more accurate approximation in 1.7 milliseconds. Consequently, this method is by far superior if fast computation times are required.

7.6 Numerical results

Numerical examples are presented in Appendix A. A total of eighteen instances are treated by comparing the different evaluation methods. These methods are: simulation, the equivalent random method, the ED method, the HF method, the IPP method, and the HyperExponential-Decomposition

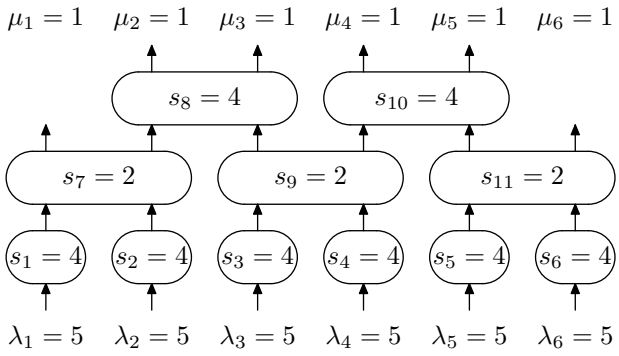


Figure 7.6: High dimensional case

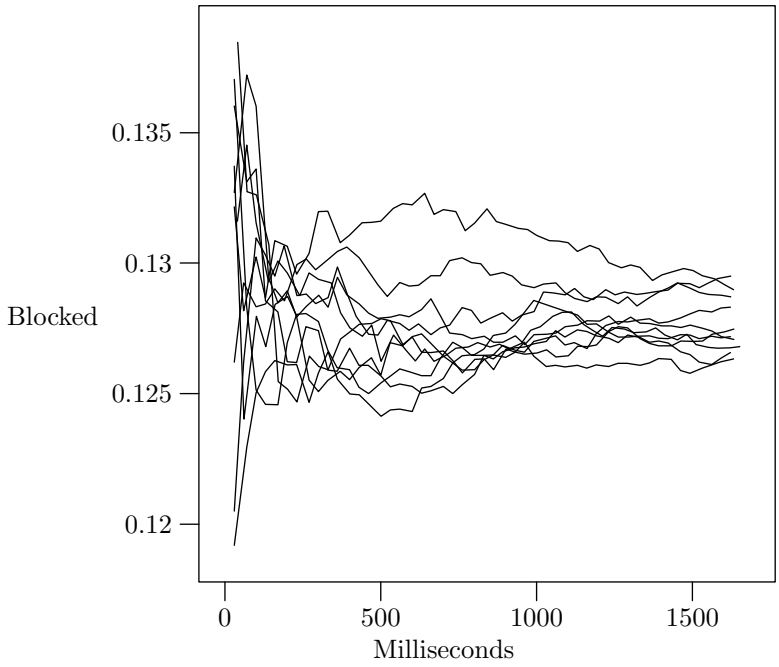


Figure 7.7: Sample paths of simulation runs

method. The blocking probabilities per skill are given in the table, as well as the weighted average. The instances are chosen in such a way that the diversity is high. The first 7 instances have equal service rates. Instances 8–14 have more variation in the service rates among the different classes. Finally, instances 15–18 contain more server groups and skills than the preceding ones.

The deviations in blocking probability with regard to simulation are presented in Table 7.1 for five different methods: the ERM, the HF method, the IPP method, the ED method and the HED method. The examples correspond to the instances of Appendix A. Concerning the ERM, in our implementation the method given by Rapp is not used, see Cooper (1981) page 170. Instead, s and a are determined from a table. As we mentioned in Section 7.1 the ERM could only be applied to a few instances, with special routing networks.

Table 7.1 shows that the IPP and the ED methods are less accurate than the HF and HED methods. The HF method is easier to implement and can handle larger server groups with more skills.

Finally, we make a short note about the weakness of the ED method that was introduced in Koole and Talim (2000). The ED method performs poorly in call centers with many layers in the overflow routing policy and many servers. See Figure 7.8 for an example. According to the ED method the blocking probability is only 0.4%, while simulation yields 4.3%. (The HED, taking higher moments into account, gives an approximation of 4.2%.) It is obvious that the inaccuracy of the ED method is caused by the fact that the distribution of the interoverflow times is assumed to be exponential. As a result of this assumption, the burstiness is underestimated. This issue was already pointed out in Koole, Pot, and Talim (2003).

7.7 Concluding remarks

The main contribution of this chapter is the introduction of a method to approximate blocking probabilities in multi-skill call centers. Our main conclusions are:

- The HED method, compared to the other methods from the literature, yields the most accurate approximations of the overall blocking probability, as well as for the individual blocking probabilities of each job type.

	Relative deviation (%)				
Method	ERM	HF	IPP	ED	HED
1	1.1	1.3	3.2	31.9	0.0
2	4.6	1.4	4.8	24.0	0.5
3	-	3.0	9.5	21.6	0.7
4	2.3	1.7	7.4	20.0	1.3
5	-	3.1	31.0	44.2	3.1
6	-	4.2	20.2	35.3	0.8
7	0.4	4.2	0.1	68.0	0.2
8	-	1.6	0.6	23.6	0.1
9	-	2.0	5.6	19.4	0.6
10	-	1.3	22.7	37.8	0.7
11	-	0.1	9.6	25.6	0.2
12	-	1.5	21.9	30.3	2.7
13	-	6.0	11.2	27.4	1.5
14	-	12.7	72.4	79.9	2.3
15	-	2.8	1.0	3.6	0.7
16	-	8.1	19.1	24.8	8.8
17	-	5.4	24.8	29.9	6.5
18	-	3.0	8.7	15.5	0.0

Table 7.1: Accuracy with regard to simulation

- Using the weighted average service rate of each group, instead of the original service rates, has not much influence on the quality of the HED method. The advantage of using weighted average service rates is a reduction of the state space, such that the balance equations are easier to solve.

- The approximation methods mentioned in this chapter, in particular the HED method and HF method, are by far superior to simulation because of the short computation times. This is discussed and illustrated in Section 7.5.

- A limitation of the HED is that call centers with relatively large groups and having many skills are hard to solve computationally. In that case the HF method is a good alternative. This lies in the fact that an exact analysis of a group in the decomposition becomes intractable when the state space grows. However, improvement of performance is possible by splitting the servers from one group into several other

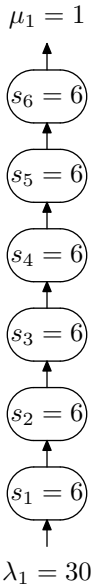


Figure 7.8: A bad case of the ED method

groups. These groups can be put behind each other in the overflow routing network, such that the system's behavior remains the same. A drawback is that the accuracy decreases somewhat, see instances 7 and 8 from Appendix A.

We list a number of subjects for future research. A possible extension to the model is to allow service rates that are group-dependent into the model. This has already been implemented in our software, but is not considered in this chapter. Service rates that are also group-dependent are in some cases very useful, of which a multi-skill call center is an example. Then agents with multiple skills are less efficient than specialists. The reason is that the work of generalists varies excessively.

Another extension would be to allow randomized routing in the HED method. We speak of randomized routing if jobs are assigned randomly to different agent groups. This extension is straightforward to support in the HED.

We mention different possibilities to further improve the HED method. In the HED method, overflow processes are split in such a way that only the

first moment of each process is correctly approximated, see Section 7.4. A possible improvement could be to split the hyperexponentially distributed overflow streams more accurately.

Interpolation is applied at several places in the HED method. We suggest to evaluate other interpolation techniques than linear interpolation in order to obtain more accurate approximations of the blocking probabilities.

Chapter 8

Calculating Staffing Levels

The subject of this chapter is the optimization of staffing levels in multi-skill call centers. The working day is split in consecutive time intervals, and the call center is considered to behave stationary in each interval. We describe a method to determine a good mix of agents having different skills for each day interval, and our objective is to minimize employee costs, while meeting service-level constraints. Service levels are measured by simulation, while optimization is done by using the HyperExponential-Decomposition method, see Chapter 7. This method is based on a blocking system with no queues, such that the performance is measured by blocking probabilities.

8.1 Introduction

Salaries of agents contribute substantially to the total costs of a call center. Therefore, it is important to schedule agents such that the cheapest and fastest agents work the most, while maximizing service-level measures or satisfying service-level constraints.

In this chapter we describe a method to determine staffing levels for multi-skill call centers with several agent groups, conform Chapter 2. The method determines a nearly optimal configuration of agents among the agent groups, which we call a staffing vector. If the working day is split into several time intervals, the method can be used for each period.

Recall that staffing is part of workforce management, which deals with the scheduling of agents. Usually, scheduling occurs on a weekly or monthly basis, such that it is important that accurate predictions of workload are available.

Finding an optimal staffing vector is a difficult problem because the number of permutations of assigning agents to groups is huge and the performance of the different permutations is difficult to compare. Besides, the performance of a multi-skill call center is hard to measure, since the performance depends on the routing policy and the skill set of each agent. Only simulation produces accurate approximations, but is time consuming. Finally, the workload is difficult to predict. A bad prediction can have a substantial impact on the service level, also expressed as "garbage in, garbage out".

With regard to relevant literature we mention Koole, Pot, and Talim (2003). They present a heuristic to optimize the division of agents among the groups. The method from this chapter is an extension because costs of agents and service-level conditions are taken into account as well. For a journal version of this work we refer to Bhulai, Koole, and Pot (2006).

Chevalier and Van den Schrieck (2005) consider overflow routing in multi-skill blocking systems with randomization parameters. Examples with four different skills are provided, and an agent group is included for each possible set of skills. They describe a method to optimize the staffing levels and randomization parameters by using Branch-and-Bound, see Land and Doig (1960).

This chapter is structured as follows. Section 8.2 describes the model and explains the objective. With regard to routing, we consider an overflow policy that is determined in advance. In Section 8.3 the algorithm that optimizes the staffing levels is presented. To this end, we first introduce an ordering of all potentially feasible staffing vectors, such that an efficient search procedure is possible. While it is assumed that routing occurs according to a predefined overflow policy, we show in Section 8.4 that even the optimization of overflow routing policies is difficult in itself and choices can have a substantial impact on service levels. In Section 8.5 the staffing procedure is applied to a few realistic situations. Finally, we state our final remarks and give a summary in Section 8.6.

8.2 Model and objective

Different types of calls are served by the call center, denoted by indices from set $\mathcal{M} := \{1, 2, \ldots, M\}$, with M the total number of job types. The work of each call type arrives according to a Poisson process with parameter λ_m

and there is a queue for each type m, with $m \in \mathcal{M}$. Calls that are not immediately served are stored in the queue that is associated with the type. We assume that agents are grouped such that agents from the same group are assumed to be identical, with respect to skills and service times. The set of agent group indices is $\mathcal{G} := \{1, 2, \ldots, G\}$ and group g has skill set S_g, $g \in \mathcal{G}$. The notation is the same as in Chapter 2. The service rate of an agent from group g serving a job of type m is μ_{mg}. It is assumed that fast and slow agents deliver the same quality of service to the customers.

The assignment of a call to an agent is governed by a routing policy, that will be denoted by π. In particular, in this chapter we consider overflow routing policies, as discussed in Section 3.4.2. This type of policy is characterized as calls being assigned to the first available agent while traversing the groups of agents in a fixed order depending on the call type. We remark that without any restriction also other types of policies can be used. For many types of routing policies, it is difficult to evaluate the service level of a given configuration of agents. No closed-form expressions exist and numerical methods suffer from the curse of dimensionality. Overflow routing is an exception because useful approximation methods have been developed, see Section 8.3 for a discussion.

In our model the allocation of agents to groups is static, i.e., we do not allow adjustments to the skills sets of the agents. However, if call centers allow that the sizes of agent groups change during the day, which is very realistic, the method can still be applied and be useful by splitting the day in different intervals and applying the method from this chapter to each interval separately.

We define a function $Q^\pi(s)$ that denotes the service level under the control policy π, with s a vector of length G denoting the staffing levels of the different groups. Our definition of service level is the percentage of callers that wait less than twenty seconds.

The staffing cost is denoted by $K(s)$ and is determined by calculating the expected cost of scheduling s_g agents in group g. Mathematically,

$$K(s) := \sum_{g=1}^{G} K^g(s_g),$$

with $K^g(s_g)$ the cost of scheduling s_g agents in group g.

The objective function is

$$f(s) := -K(s).$$

We consider optimization problems of the form

$$\begin{aligned} &\max_{s \in \mathbb{Z}_+^G} f(s) \\ &\text{subject to} \\ &Q^\pi(s) \geq \alpha, \end{aligned} \tag{8.1}$$

with $\mathbb{Z}_+$ the set of non-negative integers. Our objective is to optimize the group sizes s_g under a service-level constraint. Parameter α denotes the lowest acceptable service level. In conjunction with our service-level definition, $\alpha = 80\%$ is frequently used by the industry. Finally, during the optimization we assume a fixed overflow routing policy, and a fixed set of agent groups, as well as their skill sets.

This problem is intractable since the number of different agent configurations s is huge and methods for performance evaluation are intractable because they suffer from the curse of dimensionality. Experiments show that there does not exist a simple ordering of the staffing vectors so that simplification is easily possible.

8.3 Optimization

In the multi-skill setting with different agent groups we propose to approximate the optimal solution by a local-search heuristic. We develop an iterative algorithm that compares different staffing vectors and selects the one with the best properties. Hence, we need to specify which vectors are compared in each iteration as well as a measurement to find improvements. Both should be chosen in such a way that it is likely that the algorithm converges to optimality. A possibility is to choose the neighbor with the highest reduction of staffing costs. However, this could result in suboptimal solutions. Consider the case that there is a staffing vector with slightly higher costs and yielding a much better service level. The algorithm would ignore this vector because of the higher costs. Obviously, this is suboptimal because a higher service level could result in lower costs in later iterations.

In order to reduce the number of neighbors in the local search method, we specify an ordering of staffing vectors. First the search space is decomposed by considering only neighbors with the same total number of agents.

Reformulating (8.1) results in

$$\max_{n\in\mathbb{Z}_+}\left(\max_{s\in\mathbb{Z}_+^G:se=n} f(s)\right)$$

subject to

$$Q^\pi(s)\geq\alpha.$$

Thus, for a fixed n the remaining problem is to find the staffing vector that minimizes the employee costs while meeting the service-level constraint. Our suggestion is to combine the service level and employee costs in one objective function by using a Lagrange relaxation. The variable β is the Lagrange multiplier of the service-level constraint, representing the sensitivity of $K(s)$ to a change in α. Hence, we define

$$f(\beta,s):=\beta(Q^\pi(s)-\alpha)-K(s) \tag{8.2}$$

and

$$f^{(n)}(\beta):=\max_{s\in\mathbb{Z}_+^G:se=n} f(\beta,s).$$

We remark that we are solving a discrete optimization problem so that there can be a duality gap. Furthermore, the service level is only concave in the number of agents if the system is stable. Therefore, we ignore staffing vectors that yield unstable systems. However, a problem arises in case of abandonments, because the service level is not a concave function any more, see Koole (2003). To this end, we formulate the algorithm in such a way that we only require monotonicity of the service level in the number of agents.

By applying the Lagrange relaxation, (8.1) becomes

$$\max_{n\in\mathbb{Z}_+:n>L^n} f^{(n)} \tag{8.3}$$

with

$$f^{(n)}:=\min_{\beta\in\mathbb{R}} f^{(n)}(\beta),$$

and L^n denoting the smallest number of agents such that the service-level constraint is still satisfied. Finally, in the remainder of this section we need

to refer to the service level associated with the optimal staffing vector with n agents and Lagrange multiplier β, defined as

$$Q^{\pi}(\beta, n) := Q^{\pi}(\arg \max_{s \in \mathbb{Z}_{+}^{G} : se = n} f(\beta, s)).$$

We now present a heuristic to solve (8.3) by using different optimization techniques, which are only applicable under specific assumptions. In summary, the total number of employees n is optimized by the golden ratio search, the Lagrange multiplier β by bi-section, and the staffing vector for a fixed n and β by local search. See for example Press, Teukolsky, Vetterling, and Flannery (2002) for a description of the mentioned optimization methods. The local-search algorithm considers all possible movements of agents from one group to another and performs the movement with the highest increase in the objective function.

STAFFING HEURISTIC()

1 Initialization: $f \leftarrow 0$, $n_L \leftarrow 0$, $n_U \leftarrow \mathsf{M}'$, $\beta_L \leftarrow 0$, and $\beta_U \leftarrow \mathsf{M}'$
2 For all $n \in \mathbb{Z}$ (golden ratio search):
3 Init: $f^{(n)}(\beta) \leftarrow \mathsf{M}'$ and $\beta \leftarrow \beta_L + (\beta_U - \beta_L)/z$
4 While $\beta_U - \beta_L > \epsilon$ (bi-section)
5 Init: $f^{(n)}(\beta) \leftarrow 0$ and $s^* \leftarrow 0$ and declare $Q^{\pi}(\beta, n)$
6 For $s \in \mathbb{Z}^G : se = n$ (local search)
7 Determine neighbor s' and $Q^{\pi}(s')$ that maximizes $f(\beta, s')$
8 If $f(\beta, s') > f^{(n)}(\beta)$
9 $s^* \leftarrow s'$, $f^{(n)}(\beta) \leftarrow f(\beta, s')$, and $Q^{\pi}(\beta, n) \leftarrow Q^{\pi}(s')$
10 End if
11 Next
12 If $f^{(n)}(\beta) < f^{(n)}$
13 $s^{**} \leftarrow s^*$ and $f^{(n)} \leftarrow f^{(n)}(\beta)$
14 End if
15 If $Q^{\pi}(\beta, n) < \alpha$ then $\beta_L \leftarrow \beta$ else $\beta_U \leftarrow \beta$ End if
16 $\beta \leftarrow \beta_L + (\beta_U - \beta_L)/z$
17 Next
18 If $f^{(n)} > f$ then $f \leftarrow f^{(n)}$ and $s^{***} \leftarrow s^{**}$ End if
19 Update n_L, n_U, and n by using the golden ratio.
20 Next

The heuristic consists of several parts. The initialization occurs in lines

1–5 and uses several parameters: n_L/n_U is the lower/upper bound of the total number of agents, β_L/β_U is the lower/upper bound of β, and M' is a large number, e.g., 10^4. In lines 6–11 the heuristic finds the optimal agent configuration with respect to $f(\beta, s)$ for a fixed Lagrange multiplier β and a fixed number of agents $n = se$. Lines 15–16 optimize the costs by varying β, whereas lines 18–19 vary n. Optimality with respect to β is tested in lines 12–14, which is also checked against optimality with respect to n. The result is stored in the variable s^{***}, which is the final output of the algorithm.

A few lines require additional explanation:

1. In the part where the optimal agent configuration is determined, lines 6–11, we mention a local search algorithm that has not yet been defined. To this end, we define an ordering on the set $\{s_t : s_t e = n_t\}$ such that $s_1 < s_2$ if and only if $f(\beta, s_1) < f(\beta, s_2)$. Theoretically, the local search algorithm works well if there exists a path $s_1 = s^1 < s^2 < \ldots < s^p = s_2$ of length p such that $s^{i+1} = s^i - e_m + e_n$, with $m, n \in \mathcal{G}$ and e_i a vector with a 1 at index i, and 0 otherwise. In practice, it also performed well in all our experiments. In the local search algorithm the next state $s^{i+1} = s^i - e_m + e_n$ is the successor of s^i chosen such that $f(\beta, s^{i+1})$ is maximized over all $m, n \in \mathcal{G}$ and $f(\beta, s^{i+1}) > f(\beta, s^i)$.

2. Concerning 1., the heuristic also allows performance evaluation methods for the service level other than simulation. This is desired to reduce computation times. We refer to Section 8.3 for the alternative methods. A disadvantage of these methods is that the absolute values of the service-level approximations are not accurate. However, changes in service levels by adjustments to the sizes of the agent groups can be measured quite accurately, see Section 8.3 for a discussion. Hence, to make sure that the service-level converges to a value close to α, we did an extension in line 15. To let this work, we require $Q^{(n),\pi}(\beta)$ to be a monotonously increasing function in β. This is likely to hold because if the Lagrange multiplier of the service-level increases, the service level has a higher weight in the objective function, such that it is likely to increase. Thus, it enables us to optimize the Lagrange multiplier β such that the service level approaches α, ensuring that the service-level condition is met.

3. The second part of the heuristic deals with the optimal value of β. The implementation is based on the assumption that $f^{(n)}(\beta)$ is convex in

β for fixed n_t. It indeed appears that when β increases, the service level becomes more relevant when determining s_t, resulting in a higher value of $f^{(n)}(\beta)$ due to more conservative scheduling. The third part of the heuristic optimizes n and relies on concavity of $f^{(n)}$. Indeed by intuition, the service level is concave in n if the system is stable and the costs $K(s_t)$ will typically be linear. Notice that n is not a continuous variable such that results can be inaccurate. The variable n is optimized with the golden ratio search. In line 19 we always increase n_L if the system is not stable. This is checked by taking $\beta = M'$ and applying the local search algorithm from lines 6–10, such that the service level is maximized.

To evaluate the performance we ran several tests, see Section 9.4. Our conclusion is that this staffing algorithm works well and produces nearly optimal results in reasonable cases.

Performance improvement

With the initialization it is preferable to choose n_L and n_U as close to each other as possible. This will increase the speed of the algorithm. The Erlang-C formula might be helpful to calculate lower and upper bounds for the total number of agents. A lower bound can be obtained by considering the system as a single-skill call center and taking the lowest service rate over all groups and skills. An upper bound can be obtained by considering separate call centers for each job type.

A straightforward technique to evaluate the system in each iteration is simulation, which is often used in the industry. However, this technique is time-consuming. The calculation times can be reduced substantially by reducing the number of simulations. This can be done by also using methods from the literature that are based on a multi-skill blocking model, see for example Franx, Koole, and Pot (2006), Chevalier and Tabordon (2003), or Chapter 7. These methods assume no queues and approximate the blocking probability of the system. Although these assumptions are not made in our model, these methods from the literature are very useful; earlier studies have shown that when comparing two different staffing levels, the difference in blocking probabilities accurately reflects the difference in service levels for the original system, see Koole, Pot, and Talim (2003) for a first example, and Chevalier, Shumsky, and Tabordon (2004) and Chevalier and Van den Schrieck (2005) for a method that integrates one of these methods in an

optimization algorithm. Thus, these methods are useful for optimization purposes, and therefore used in all of our numerical examples. We conclude from our experiments that they reduce the computation times considerably. To support blocking models, the algorithm is modified as follows:

- replace in lines 6–11 Q by $\widetilde{Q}$, denoting 1 minus the blocking probability instead of the probability of waiting less than twenty seconds,
- replace β by $\tilde{\beta}$ because the Lagrange multiplier has no longer the same meaning,
- redefine in Equation (8.2) $f(\beta, s) = -K(s)$, and
- determine after line 11 $Q^{\pi}(s^*)$ by simulation, and calculate $f^{(n)}(\beta)$.

Line 15 is of crucial importance because it ensures that the service-level constraint is met. For that reason, the redefinition of $f(\beta, s)$ is justified.

Wallace and Whitt (2005) show numerically that when each agent has at most two skills, the performance can be almost as good as when each agent has all skills. This result is relevant to step 4 of the labor allocation process. Scheduling agents in groups such that they use only one or two skills simplifies step 4 of the labor allocation process. It reduces the probability of having an infeasible solution in step 4 and possibly also improves the quality of the resulting rosters.

8.4 Overflow routing

Overflow routing is a type of policy that is often used by the industry, see Chapter 7. In this chapter we assume that the overflow routing policy is determined in advance and is not optimized during the staffing algorithm. This might suggest that the routing policy hardly influences the performance. However, this is certainly not the case; the parameters of the overflow policy can have substantial influence on the service level, and it is difficult to find good or optimal parameters.

In this section we show by means of examples that simple rules, that one would expect by intuition, only apply under certain assumptions, see for example the theorems from Chapter 4. For simplicity, queues are omitted and the service level of a call type is calculated as the percentage of callers that are not blocked. The overall service level is defined as the weighted

average of the service level of each type, with the weight of type m being calculated by $w_m\lambda_m$. We refer to Chapter 2 for a further description of the weights w_m. The distributions in the examples are all exponential.

Counter examples

As one would expect, overflow routing policies are not optimal. This is shown in the following example.

Example 1: Consider a service center defined as: $M = 3$, $G = 2$, $S_1 = \{1, 2\}$, $S_2 = \{2, 3\}$, and the routing policy is to assign calls to group 1 if possible, and otherwise to group 2, as depicted in Figure 8.1. It is clear that

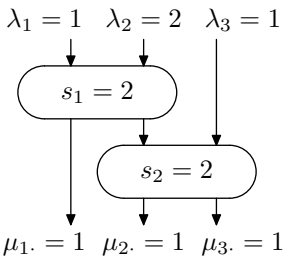


Figure 8.1: Service center from Example 1

this setting is not optimal, because jobs of class 2 are always assigned first to agents of group 1. This can result in numerous rejections of jobs from class 1 because the occupancy of this group is high. The performance increases by assigning jobs of type 2 to agents of groups 1 and 2 in the same proportions randomly, so the number of rejected jobs from class 1 decreases. We illustrate this by means of a numerical example. Taking parameters $\lambda_1 = \lambda_3 = 1$, $\lambda_2 = 2$, $\mu_{11} = \mu_{22} = \mu_{13} = \mu_{23} = 1$, $s_1 = 2$, $s_2 = 2$, $w_1 = w_3 = \frac{1}{4}$, and $w_2 = \frac{1}{2}$ (notice that the service level is symmetric) results in:

1. 64.78% service level with non-randomized overflow routing,
2. 65.20% service level with randomized overflow routing, and
3. 65.31% service level with optimal dynamic routing.

This shows that overflow routing policies are not always optimal.

The next examples show that properties of optimal policies that sound plausible, do not always apply.

Example 2: There are cases with equal λ's, μ's and w's in which replacing a generalist by a specialist leads to a higher service level. Consider a service center with three skills and three arrival processes, denoted by $M = 3$. We assume that $\mu_{31} = \mu_{12} = \mu_{22} = \mu_{23} = \mu_{33} = 1$, $\lambda_1 = \lambda_2 = \lambda_3 = 3$ and $w_1 = w_2 = w_3$. In addition, $G = 3$, $S_1 = \{3\}$, $S_2 = \{2, 3\}$, $S_3 = \{1, 2\}$, and the routing policy is depicted graphically, see Figure 8.2. We compare two

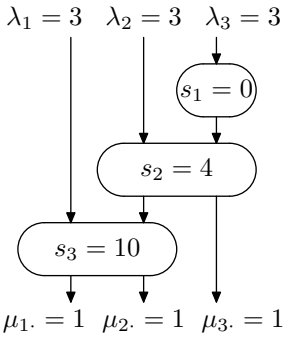


Figure 8.2: Service center from Example 2

situations with different sizes of agent groups:

1. $s_1 = 0$, $s_2 = 4$ and $s_3 = 10$ yields a service level of 84%, and
2. $s_1 = 3$, $s_2 = 1$ and $s_3 = 10$ yields a service level of 89%.

The difference between situations 1 and 2 is that in situation 2 three agents are moved from group 2 to group 1, such that specialists are dedicated to type 3. In situation 2 the agents from group 1 only handle jobs from class 3. Consequently, fewer jobs from class 3 are rejected. The service level of jobs from classes 1 and 2 will hardly decrease because sufficient agents are available in group 3, whereas the service of jobs from class 3 increases substantially.

Example 3: In this example we show that the service level can increase by replacing a generalist by a specialist upstream, which even holds when the service levels of both types have equal weights. Consider a service center with two skills and two arrival processes, denoted by $M = 2$. We assume

that $\mu_{11} = \mu_{13} = \frac{2}{10}$ and $\mu_{22} = \mu_{23} = 2$, $\lambda_1 = \lambda_2 = 1$ and $w_1 = w_2 = \frac{1}{2}$. In addition, $G = 3$, $S_1 = \{1\}$, $S_2 = \{2\}$, $S_3 = \{1, 2\}$. Figure 8.3 specifies the routing policy. We will compare two situations with different sizes of agent

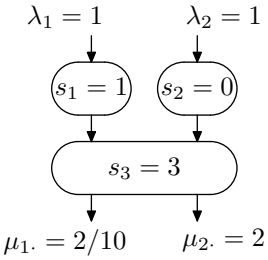


Figure 8.3: Service center from Example 3

groups. These are:

1. $s_1 = 1$, $s_2 = 0$ and $s_3 = 3$, with a service level of 54%, and
2. $s_1 = 1$, $s_2 = 1$ and $s_3 = 2$, with a service level of 62%.

The difference between both situations is that in the second case one agent is less skilled and dedicated to type 2, in comparison to situation 1. In this way more calls are served because the service rate of type 2 is higher.

Example 4: We show that replacing a generalist by a specialist in front can lead to a higher service level, even when the service rates are equal. Consider a service center with two skills and two arrival processes, denoted by $M = 2$. We assume that $\mu_{11} = \mu_{22} = \mu_{13} = \mu_{23} = 1$, $\lambda_1 = \lambda_2 = 1$, $w_1 = 0.1$ and $w_2 = 0.9$. In addition, $G = 3$, $S_1 = \{1\}$, $S_2 = \{2\}$, $S_3 = \{1, 2\}$, with a complete specification depicted in Figure 8.4. This time we compare

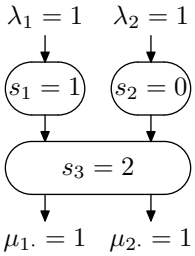


Figure 8.4: Service center from Example 4

the following two situations:

1. $s_1 = 1$, $s_2 = 0$ and $s_3 = 2$, yielding 71% service level, and

2. $s_1 = 1$, $s_2 = 1$ and $s_3 = 1$, yielding 73% service level.

Jobs from class 2 have a much higher weight than jobs from class 1. The weighted average service level is increased by minimizing the number of rejected jobs from class 2. Therefore situation 2 is preferred because one agent is available all the time to serve jobs from class 2. This is an improvement because the service times of type 2 are stochastically smaller in comparison to type 1.

Example 5: In this example we show that under certain overflow policies the service level can decrease when an agent is replaced by a strictly more skilled agent downstream in the overflow network. However, in general one would expect that placing a specialist after a generalist in the overflow network would be bad. Consider again a service center with two skills and two arrival processes, denoted by $M = 2$. We assume that $\mu_{11} = \mu_{21} = \mu_{22} = 1$, $\lambda_1 = \lambda_2 = 1$, $w_1 = 0.1$ and $w_2 = 0.9$. In addition, $G = 2$, $S_1 = \{1, 2\}$, $S_2 = \{2\}$, and please look at Figure 8.5 for the remaining details. Consider

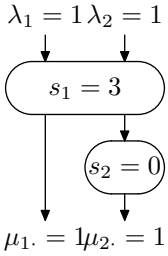


Figure 8.5: Service center from Example 5

the following two situations:

1. $s_1 = 3$ and $s_2 = 0$ results in a service level of 79%, and

2. by taking $s_1 = 2$ and $s_2 = 1$ we obtain a service level of 83%.

If we had placed the specialist in front of the generalists, the improvement would have been smaller.

Example 6: It is not always optimal to route jobs to one-skilled specialists first, even when the service level is symmetric and all μ's are equal. To

show this we will consider two situations that correspond to a call center with two skills and two arrival processes, denoted by $M = 2$. In addition, $\lambda_1 = \lambda_2 = 1$, $w_1 = w_2 = \frac{1}{2}$, $\mu_1 = \mu_2 = \frac{1}{2}$, $G = 3$, $S_1 = \{1\}$, $S_2 = \{1\}$, $S_3 = \{1, 2\}$, $S_4 = \{2\}$, see Figure 8.6. Next, we compare the following two

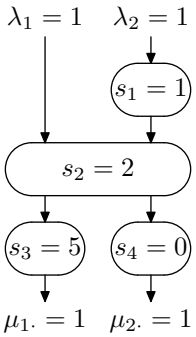


Figure 8.6: Service center from Example 6

situations:

1. $s_1 = 1$, $s_2 = 2$, $s_3 = 5$ and $s_4 = 0$ yields a service level of 91.2%, and
2. $s_1 = 0$, $s_2 = 2$, $s_3 = 5$ and $s_4 = 1$ yields a service level of 92.9%.

Note that almost all jobs of type 1 are served because of the relatively high number of agents in group 3. In the second scenario, the agents of group 2 serve additional jobs of type 2, compared to scenario 1. The reason remains that the overflow rate of type-2 jobs to group 2 is higher. This increases the service level of type 2, while the service level of type 1 remains almost unchanged.

8.5 Examples

The effectiveness of the algorithm from Section 8.3 is illustrated by means of two examples, of a two- and three-skill call center. Although the call centers are of moderate size the algorithm can easily be applied to larger ones with acceptable computation times.

The models of the examples contain queues with an infinite length, customers having infinite patience for waiting, and jobs of a certain type are served according to a first-come-first-serve service discipline. The calls are assigned to the agents according to overflow policies, see for example Franx,

n								
16			17			18		
$\tilde{\beta}$	s^*	Q^π	$\tilde{\beta}$	s^*	Q^π	$\tilde{\beta}$	s^*	Q^π
2.5_3	(6,4,6)	80%	2.5_3	(7,5,5)	87%	2.5_3	(8,5,5)	93%
6.3_2	(7,4,5)	80%	6.3_2	(8,4,5)	88%	6.3_2	(9,4,5)	93%
1.6_2	(7,4,5)	80%	1.6_2	(8,4,5)	88%	1.6_2	(9,4,5)	93%
3.9_1	(9,5,2)	81%	3.9_1	(10,5,2)	87%	3.9_1	(10,6,2)	91%
9.8_0	(11,5,0)	76%	9.8_0	(11,6,0)	84%	9.8_0	(12,6,0)	89%
1.7_1	(10,6,0)	76%	2.4_0	(14,3,0)	41%	2.4_0	(15,3,0)	42%
2.3_1	(10,6,0)	76%	4.3_0	(12,5,0)	81%	4.3_0	(13,5,0)	83%
2.7_1	(10,6,0)	76%	-	-	-	-	-	-
3.6_1	(9,5,2)	81%	4.3_0	(12,5,0)	81%	4.3_0	(13,5,0)	83%
	9.8			9.5			10	

Table 8.1: Output of the staffing algorithm (with two skills)

Koole, and Pot (2006), and in such a way that specialists have the highest priority, agents with two skills the second-highest priority, and agents with three skills the third-highest priority. At a service completion, a job from the longest queue is taken, among the queues of jobs for which the agent has the required skill.

Two skills

The algorithm is applied to a call center with two skills and three agent groups, $G = 3$ and $\mathcal{M} = \{1, 2\}$. Figure 5.1 shows an example in case that $M = 2$. The arrival rates are $\lambda_1 = 1.5$ and $\lambda_2 = 2$. A group of specialists is included for each call type, $S_1 = \{1\}, S_2 = \{2\}$, and additional flexibility is ensured by generalists, $S_3 = \{1, 2\}$. The costs c_g of staffing an agent in the g-th group are 0.5 if $g = 1$, 0.7 if $g = 2$, and 0.9 in case $g = 3$. Specialists are more efficient than generalists, $\mu_{11} = 0.18$, $\mu_{22} = 0.6$, $\mu_{13} = 0.16$, and $\mu_{23} = 0.5$.

The algorithm requires a few parameters before execution. Hence, we choose $\mathsf{M}' = 10^4$, $n_L = 15$, and $n_U = 20$. The bi-section parameter is 4 instead of 2, such that per iteration either one quarter or three quarters of the search space is excluded. This reduces the number of iterations, which makes it easier to present the results in the tables.

Table 8.1 summarizes the main results that are obtained by applying the

algorithm to this example. The table consists of three columns, each for a different number of agents n, determined by the optimization procedure. For each n it shows in the rows the $\tilde{\beta}$ obtained by bi-section, together with the optimal staffing level obtained by local search, and the corresponding service level. The variable $\tilde{\beta}$ converges over the different rows to the optimal value. The value of the subscript k in the column of $\tilde{\beta}$ denotes that the value should be multiplied by 10^k. The table only presents data for the first 9 iterations of the bi-section algorithm because these show the most significant improvements.

The last two rows show the optimal outcome, for each value of n; the first row presents the optimal $\tilde{\beta}$, Q^τ, and s^*, the second row contains the staffing costs of the optimal vector. In addition, the staffing costs $K(s^*)$ are 9.8, 9.5, and 10 for 16, 17, and 18 agents, respectively. Furthermore, taking $n = 19$ yields as optimum $K(s^*) = 10.5$. The objective value of 9.5 for $n = 17$ is the best solution found. It also appeared to be optimal; no better solution was found by enumerating the whole search space and evaluating each staffing vector.

The algorithm requires the simulation of a few more than seventeen different staffing vectors, taking a few seconds computation time each. Besides that, the optimization procedure performed by local search hardly takes any time.

Three skills

Adding a skill to the set of the previous example increases the number of different skill sets from 3 to 7. In case of three skills, the skills of the agent groups are: $S_1 = \{1\}$, $S_2 = \{2\}$, $S_3 = \{3\}$, $S_4 = \{1, 2\}$, $S_5 = \{2, 3\}$, $S_6 = \{1, 3\}$, and $S_7 = \{1, 2, 3\}$. Jobs arrive with rates $\lambda_1 = \lambda_2 = \lambda_3 = 2$ and the service rates μ_{mg}, denoting the service rate of agents from group g working on call type m, are: $\mu_{11} = 0.3$, $\mu_{22} = 0.4$, $\mu_{33} = 0.5$, $\mu_{14} = 0.27$, $\mu_{24} = 0.36$, $\mu_{25} = 0.36$, $\mu_{35} = 0.45$, $\mu_{16} = 0.27$, $\mu_{36} = 0.45$, $\mu_{17} = 0.24$, $\mu_{27} = 0.32$, and $\mu_{37} = 0.40$. The costs of the agents are: $c_1 = 1.3$, $c_2 = 1.4$, $c_3 = 1.5$, $c_4 = 1.4$, $c_5 = 1.6$, $c_6 = 1.5$, and $c_7 = 1.6$.

A summary of the execution of the algorithm is depicted in Table 8.2. The structure of the table is similar to Table 8.1, and will not be explained in further detail.

The table shows that no feasible solution was found for $n = 21$; even for very high values of $\tilde{\beta}$ the service level is below 78%, while 80% is required.

n												
21			22			23			24			
$\tilde{\beta}$	s^*	Q^π	$\tilde{\beta}$	s^*	Q^π	$\tilde{\beta}$	s^*	Q^π	$\tilde{\beta}$	s^*	Q^π
2.5_3	(4,3,2,2,4,2,4)	78%	2.5_3	(4,3,2,2,3,2,6)	86%	2.5_3	(5,3,2,1,4,2,6)	91%	2.5_3	(4,2,1,3,4,2,8)	95%
4.4_3	(4,3,2,2,4,2,4)	78%	6.3_2	(4,3,2,2,3,2,6)	86%	6.3_2	(5,3,2,2,3,2,6)	91%	6.3_2	(5,4,2,2,2,3,6)	95%
5.8_3	(4,3,2,2,4,2,4)	78%	1.6_2	(5,4,3,3,1,2,4)	86%	1.6_2	(5,4,3,4,1,2,4)	91%	1.6_2	(5,3,3,4,1,2,6)	95%
6.8_3	(4,3,2,2,4,2,4)	78%	3.9_1	(6,5,4,3,0,2,2)	85%	3.9_1	(6,4,3,4,0,2,4)	91%	3.9_1	(7,4,4,5,0,2,2)	93%
7.6_3	(4,3,2,2,4,2,4)	78%	9.8_0	(9,5,5,3,0,0,0)	75%	9.8_0	(10,6,5,2,0,0,0)	78%	9.8_0	(10,6,4,2,0,2,0)	85%
8.7_3	(4,3,2,2,4,2,4)	78%	1.7_1	(8,5,4,3,0,2,0)	80%	1.7_1	(8,4,4,5,0,2,0)	85%	2.4_0	(19,4,1,0,0,0,0)	34%
9.0_3	(4,3,2,2,4,2,4)	78%	1.2_1	(9,5,5,3,0,0,0)	75%	1.2_1	(9,6,4,2,0,2,0)	83%	4.3_0	(14,5,3,2,0,0,0)	57%
9.2_3	(4,3,2,2,4,2,4)	78%	1.3_1	(8,5,4,3,0,2,0)	80%	1.0_1	(10,6,5,2,0,0,0)	78%	5.6_0	(13,5,4,2,0,0,0)	57%
9.4_3	(4,3,2,2,4,2,4)	78%	1.2_1	(9,5,5,3,0,0,0)	75%	1.1_1	(9,6,4,2,0,2,0)	83%	6.7_0	(11,6,5,2,0,0,0)	79%
9.6_3	(4,3,2,2,4,2,4)	78%	1.2_1	(8,5,4,3,0,2,0)	80%	-	-	-	7.4_0	(11,6,5,2,0,0,0)	79%
9.7_3	(4,3,2,2,4,2,4)	78%	-	-	-	-	-	-	8.0_0	(11,6,5,2,0,0,0)	79%
-	-	-	1.2_1	(8,5,4,3,0,2,0)	80%	1.1_1	(9,6,4,2,0,2,0)	83%	9.8_0	(10,6,4,2,0,2,0)	85%
	31.0			30.6			31.9			33.2	

Table 8.2: Output of the staffing algorithm (with three skills)

The best solution is obtained by taking $n = 22$, which resulted in a service level of 80% and staffing costs of 30.6. Higher values of n resulted in higher costs.

8.6 Concluding remarks

In this chapter we provided a description of an effective and efficient algorithm for the computation of staffing levels, based on estimates of the workload.

A possible extension is to model constraints on the sizes of the agent groups. This can easily be added to the algorithm by including bounds on the group sizes during the local search.

A useful extension to the algorithm would be to model different weights on the service level for different job types, for example by calculating weighted averages of service levels. However, the performance optimization in the method explained in this chapter is achieved in conjunction with methods for blocking systems, like for example the HyperExponential-Decomposition method. These methods are not easy to modify for different weights among job types, because approximating a weighted average of probabilities on the waiting time is not comparable to an average of blocking probabilities of a system without queues. The usage of these evaluation methods can be circumvented by performing local search in combination with simulation. However, a drawback is that computation times become extremely long.

Chapter 9

Shift Scheduling

This chapter describes methods to determine shifts and to generate schedules in multi-skill contact centers.

9.1 Introduction

Obtaining a good schedule is a difficult problem because for the following reason: a large portion of the contact centers can be classified as inbound, i.e., jobs are initiated by customers. This type of contact center is characterized by jobs that arrive randomly over time such that the workload is hard to predict; it is influenced by external factors such as the weather for example. This makes it hard to find a good match between the expected workload and the available labor resources.

Shift scheduling is an important step in the allocation of labor resources over time, which is part of workforce management. Recall that labor allocation is typically an operational problem with a time horizon of only a few weeks and it is common to distinguish four phases in the process of labor allocation:

1. workload prediction,
2. staffing,
3. shift scheduling, and
4. rostering.

Workload prediction is concerned with the prediction of the future amount of work offered to the call center. Staffing translates this amount of work

into the numbers of required agents such that the service levels are met. Shift scheduling is the generation of shifts such that the staffing levels are met. Rostering refers to the pairing of shifts into rosters and the assignment of the employees to the rosters.

It is important to find a good match between the predicted workload and the amount of scheduled workforce. An insufficient workforce can lead to low service levels, for example, too long waiting times. This can be avoided by scheduling sufficient employees. However, it is undesirable to schedule too many employees because, besides service levels, contact centers also have to meet economical objectives, in particular minimizing costs, amongst which employee salaries. Minimizing the number of employees is an important problem because labor is expensive; about eighty percent of operating costs in call centers are personnel costs, see Gans, Koole, and Mandelbaum (2003). Therefore, the cost reductions obtained from good scheduling algorithms can be substantial.

However, optimal labor allocation in single-skill call centers is a complicated problem, since the four steps in the process of labor allocation are difficult to integrate. Multi-skill call centers bring additional complexity to the above steps because agents handling jobs require different skills. With regard to labor allocation, the predicted workload is often specified per job type or skill. Hence, the determination of the staffing levels is more complicated as compared to single-skill call centers where the workload is specified by a single number.

In this chapter we deal with steps 2 and 3 in the process of labor allocation. We note that these phases have conflicting objectives. Meeting the service level can be easily realized by adding agents, however, adding agents increases personnel costs considerably. On the other hand, scheduling too few agents leads to lower personnel costs but may compromise service-level constraints. The challenge is to express both components in one criterion function because of the difficulty to compare the two quantities. In the literature this problem is usually circumvented by minimizing the personnel costs subject to a constraint on the service level.

Our main contribution is a method to determine a schedule in multi-skill call centers such that a rough match between predicted workload and labor capacity is obtained, taking the randomness of the arrival processes into account. It consists of two steps: phases 2 and 3 of the labor allocation process. The reason to solve both steps separately is that an integrated approach requires calculations that are very time-consuming to execute. In

practice, obtaining good rosters often requires several iterations between the different phases. In these cases, it is important to have a scheduling and rostering method with short computation times. The possible drawbacks of solving both steps separately will be discussed in Section 9.6. Furthermore, in the first step the service-level constraints are only included to a limited extent. For example, service levels can not be specified per job type. In our opinion, this is not a major restriction. Schedules are most often generated at least a few weeks ahead of time, based on predictions. Hence, call centers often have to reschedule during the day when the real workload deviates from the predictions. For that reason, service levels need to be controlled during operations. Besides rescheduling, another way to control the service levels is by means of optimization of the routing policies, see Section 6.

The method being discussed determines the staffing levels in phase 2 by applying the heuristic from Chapter 8. For the shift scheduling (phase 3) we present a model that generates a space of feasible solutions so that integer programming methods can be used to obtain a set of optimal shifts. The model encapsulates the flexibility of multi-skilled agents to work in different groups, using a subset of their skills in different periods. The method has small computational requirements such that it is much faster than existing methods from the literature.

In this chapter the integer programming model of the shift-scheduling method is developed for call centers. However, it is also applicable to other service systems than call centers. In general it can solve shift-scheduling problems in organizations that:

- distinguish multiple skills,
- allow employees to work consecutively on different tasks, and
- have employees with identical productivity within the same skill group.

An example is the scheduling of nurses in hospitals. It is likely that staffing levels are expressed similarly as in call centers. For example, by choosing the staffing levels in each period in such a way that the workload is covered as accurately as possible. It is realistic that some nurses use only one skill to obtain a high productivity, while others have several skills to minimize the total number of nurses. In addition, the physical location of the different tasks can play a role. If the distance between the location of two tasks is large, it is unattractive to schedule the same employee on these tasks.

Literature

The literature offers different models and algorithms for shift scheduling in single-skill call centers. However, not much literature is available about multi-skill call centers. The most relevant papers about scheduling in call centers are discussed next.

Most models are based on the standard set-covering model as presented in Dantzig (1954). This model finds an optimal set of shifts in a multi-period and single-skill environment, while obeying the service-level constraint in each period. A cost is associated with each shift and the objective is minimization of the total costs.

Keith (1979) extended the set-covering model with slack and surplus variables. This model permits fewer or more than the desired number of agents to be scheduled. This creates a balance between the cost of deviating from the desired number of agents and the reduction in the number of scheduled shifts while satisfying the service-level constraint.

Thompson (1997) introduced two models for shift scheduling. He distinguishes minimum acceptable service levels per period and a constraint on the average service level over the planning horizon. An integer programming model is described that includes both types of service-level constraints. It solves the staffing problem and the shift-scheduling problem in an integrated fashion. The paper also gives an extensive overview of the literature and makes a classification of the different shift-scheduling models. In Section 9.2.2 we give a short description of the first model to obtain lower bounds for more complicated models.

Ingolfsson, Cabral, and Wu (2002) focus on exceptional cases in which the traditional methods perform badly due to transient effects. This is typically the case with long service times because they create dependence between consecutive periods. Think of an arriving customer that will have a higher probability to stay in the system during multiple periods, because of longer service times. Hence, if the expected workload drops, the traditional methods tend to underestimate the required number of agents, since they neglect the customers in service. In addition, the paper introduces a method for staffing and scheduling in single-skill call centers.

In Atlason, Epelman, and Henderson (2004a), Atlason, Epelman, and Henderson (2004b), the model of Thompson (1997) is modified for cases in which the service level can only be obtained via simulation. The extra benefit of simulation is that it allows for calculating the service level in

a transient setting. Since simulation is a very time-consuming operation, the methods incorporate the conditions on the service level differently and more efficiently. Certain constraints are removed and handled by means of cutting-plane techniques.

Cezik and L'Ecuyer (2005) describe a generalization of the method from Atlason, Epelman, and Henderson (2004a) in the context of multi-skill call centers. The method reduces the solution space by means of generating cuts. The computation time of this algorithm is relatively long because each cut requires the multi-skill call center to be simulated multiple times and very accurately. Hence, they are not able to solve the shift-scheduling problem, but only to determine the staffing levels, that are constant during the day. We remark that the computation time is important because phase 4 (rostering) is an iterative procedure in which phase 3 is executed several times with small adjustments, e.g., at times where the workload differs from predictions. Thus, for practical purposes it is desirable to have an algorithm that executes phases 2 and 3 relatively fast.

We remark that there does not exist an easy way to extend the methods from the literature to shift scheduling in multi-skill call centers, in such a way that the resulting methods are numerically tractable.

Content

This chapter is organized as follows. Section 9.2 treats existing methods from the literature for shift scheduling in single-skill call centers that are useful to this chapter. In Section 9.3 an efficient method is presented for shift scheduling in the multi-skill environment when considering service-level constraints in each period. The method consists of two steps: a heuristic for the determination of staffing levels (see Section 9.3.1 and Chapter 8), and the determination of an optimal set of shifts (see Section 9.3.2). Next, Section 9.3.3 describes an extension to take constraints on the average service level during the day into account. The method from this chapter for scheduling in multi-skill call centers is evaluated numerically by several experiments and a case study, presented in Section 9.4. We show that the method yields nearly optimal results. In Section 9.5 we treat different extensions to the models, which are relevant for the single-skill as well as the multi-skill methods. Finally, a summary of the results and directions for future research are given in Section 9.6.

9.2 Single-skill environment

This section treats two scheduling models from the literature for single-skill call centers. The purpose of this section is twofold. In the first place it gives an introduction to the topic of shift scheduling. In the second place the methods are useful for calculating lower bounds for the objective function with respect to the multi-skill environment.

The papers that we discuss on single-skill call centers have in common that the time horizon is split in several time intervals. It is assumed that the periods are mutually independent and that the system is in equilibrium in each period. There are T periods or time intervals and the indices of the periods are specified by the set $\mathcal{T} := \{1, 2, \ldots, T\}$. Calls arrive according to a Poisson process with parameter λ_t in period t, and we assume that the arrival rate is constant in each period, with $t \in \mathcal{T}$. The service time distribution is assumed to be exponential, with parameter μ_t in period t. Note that time dependence of the service rate is not very important because in reality the service rate is often almost constant during the day. The number of shift types is fixed and denoted by K. Each shift type has an index and the corresponding indices are enclosed in the set $\mathcal{K} := \{1, 2, \ldots, K\}$. Each shift has an offset, which is denoted as the index of the starting period, and a length. However, additional characteristics, e.g., breaks and splitted shifts, are also easy to include. The system incurs a cost whenever a specific shift is used, and the objective is to select a set of shifts such that the service levels in each period are met at minimum costs. The optimal shift schedule can be obtained from the following two methods.

9.2.1 Two-step method

This section deals with the basic set-covering model as introduced in Dantzig (1954). It is applied to call centers in Keith (1979) and revisited next. The objective is to schedule the shifts such that conditions are satisfied on the minimum service level per period (instead of one condition on the average service level during the whole day). The method consists of two steps: staffing level calculation and shift scheduling. The staffing levels, as determined in the first step, are chosen such that the minimum service level during each period is assured. In the second step, shifts are scheduled such that the staffing level in each period is met.

We assume that the system is in equilibrium in each period, however, in

practice this is not the case, creating dependence between the periods. To overcome this problem one can adjust the staffing levels determined in the first step before step 2 is carried out. In the following subsection we will also describe how to achieve this.

Staffing levels

We assume that the expected workload has already been determined in step 1 of the labor allocation process. The first step of the two-step method consists of the determination of the staffing levels. For each period t we calculate the number of agents s_t that is required to meet the service-level conditions. We define the service level as the percentage of arrivals that wait less than the acceptable waiting time (AWT) in the queue. The minimum requirement is denoted by α. In practice, call centers often take $\alpha = 80\%$ and an AWT of twenty seconds.

There exist different methods to execute this step. For example, call centers determine the staffing levels often as follows. The working day is split in consecutive time intervals. Then call centers assume independence between the different periods as well as stationary behavior. Next, using standard formulas, like the Erlang C, the staffing level is determined for each time interval. (For literature about the multi-server queue and the determination of staffing levels, we refer to Cooper (1981).)

The impact of assuming independence between the time intervals can be reduced. Thompson (1993) describes a method that corrects for the multi-period impact by means of adjusting the arrival rate appropriately in each period. Green, Kolesar, and Soares (2001) propose the lag-max algorithm that shifts the staffing levels forward in time that were determined in the first step. This results in service levels that are satisfied for almost any period when taking transient effects into account.

Shift scheduling

In this subsection we describe Dantzig's model for shift scheduling. We assume that the staffing levels during the day are given by the first step. In the second step, the optimal number of shifts is calculated per shift type. A shift is specified by the working hours and breaks. To formulate the model we need the following notation:

- $a_{k,t}$ is set to 1 if shift k overlaps time interval t and 0 otherwise, with $k \in \mathcal{K}$,
- x_k denotes the number of agents working shift k,
- c_k is the cost associated with an agent working shift k, and
- s_t is the number of agents required during interval t.

To meet the service-level conditions we require that the schedule consists of a set of shifts such that at least s_t agents work in each period t.

The integer programming model to compute the optimal schedule is

$$\min \sum_{k \in \mathcal{K}} c_k x_k$$

subject to

$$\sum_{k \in \mathcal{K}} a_{k,t} x_k \geq s_t, \qquad \forall t \in \mathcal{T},$$

$$x_k \geq 0 \text{ and integer}, \quad \forall k \in \mathcal{K}.$$

The model can be formulated in a more compact form by using matrix notation, in such a way that the elements $a_{k,t}$ form a matrix A. The matrix A can be very large due to the high number of different shifts. In realistic cases, the model is tractable and has small computation times.

9.2.2 Integrated Method

The two-step method from the previous section is not optimal for certain types of service-level constraints, for example conditions on the (weighted) average service level during a day. To obtain an optimal solution, one can integrate the determination of staffing levels and shift scheduling by constructing an integer programming model. For call centers with one skill, an appropriate integer programming model is presented in Thompson (1997). The complexity is higher compared to the model of Dantzig (1954), but is computationally still tractable. Ingolfsson, Cabral, and Wu (2002) worked

on a similar subject. They describe a heuristic method for shift scheduling that takes transient effects into account.

The model from Thompson (1997) also easily extends to a multi-skill environment, which will be discussed in Section 9.4.2. This requires a modification of the integer programming formulation. In this section we discuss the modified model in case of one skill. The extension to multiple skills is straightforward. The model requires the following additional notation. We introduce parameter $\gamma_{s,t}$ denoting the expected number of customers that wait less than the AWT in the queue during interval t when we schedule s agents, i.e., λ_t times the service level in period t when s agents are scheduled in that period. The binary decision variable $n_{s,t}$ denotes 1 if there are s agents scheduled during interval t, and 0 otherwise. A constraint is included to ensure that in each interval t exactly one of these variables takes the value 1. Furthermore, we let $s \in \mathcal{S} := \{0, 1, \ldots, S\}$, and take S sufficiently high.

$$\min \sum_{k \in \mathcal{K}} c_k x_k$$

subject to

$$\begin{aligned}
&\sum_{k \in \mathcal{K}} a_{k,t} x_k = \sum_{s \in \mathcal{S}} n_{s,t} s, && \forall t \in \mathcal{T}, \\
&\sum_{s \in \mathcal{S}, t \in \mathcal{T}} n_{s,t} \gamma_{s,t} \geq \alpha (\sum_{t \in \mathcal{T}} \lambda_t), && \\
&\sum_{s \in \mathcal{S}} n_{s,t} = 1, && \forall t \in \mathcal{T}, \\
&x_k \geq 0 \text{ and integer}, && \forall k \in \mathcal{K}, \\
&n_{s,t} \in \{\, 0, 1\}, && \forall s \in \mathcal{S}, \forall t \in \mathcal{T}.
\end{aligned}$$

With respect to multi-skill call centers this model is useful. It enables us to compute lower bounds for the criterion function.

9.3 Multi-skill environment

This section introduces two methods for shift scheduling in multi-skill call centers. The methods consist of phases 2 and 3 of the labor allocation

process, see Section 9.1. The first method has similarities with the two-step method from Section 9.2 because both steps are executed separately. The second method describes a heuristic that iterates between both steps and is able to handle service-level conditions measured as the average during a day.

The major difference in the two-step method from Section 9.2 is the presence of multiple agent groups with different skills. We assume that agents from the same group have an equal set of skills. Our objective is to meet the service-level constraints at minimum costs. In the first step, a minimum staffing level is determined such that the service-level constraints are satisfied, i.e., the fraction of calls (among all types) that have an AWT of less than twenty seconds is greater than or equal to α. The staffing levels denote the required number of agents in each agent group for each period. This scheduling problem is significantly more difficult in comparison to scheduling in single-skill call centers. In multi-skill call centers the service level is influenced by the routing policy and, for that reason, the routing policy needs to be taken into account.

In the second step, a set of shifts is composed that minimizes the costs and satisfies the required staffing levels. This step is also more complicated than in a single-skill environment. In a multi-skill environment an agent with a specific set of skills can be assigned to different agent groups with potentially fewer skills in each period. Modeling this in a straightforward way leads to many decision variables, which easily results in intractable models.

Before presenting the two methods, we define the multi-skill model as follows. We consider a call center that handles calls that require a skill from the set $\mathcal{M} := \{1, 2, \ldots, M\}$. Calls of type $m \in \mathcal{M}$ arrive in period $t \in \mathcal{T}$ according to a Poisson process with rate $\lambda_{m,t}$. Every agent in the call center belongs to an agent group, that can be different in each period, from the set $\mathcal{G} := \{1, 2, \ldots, G\}$. The service rates are skill- and group-dependent, denoted by rate $\mu_{m,g}$ for skill $m \in \mathcal{M}$ and group $g \in \mathcal{G}$. We assume that a control policy π is given that defines a call-selection and agent-selection rule. Call assignment occurs according to the agent-selection rule. If a call is not assigned to an agent group, it is queued, after which it is served according to some call-selection rule.

A shift is defined by a collection of working hours from $\mathcal{T}$ and a subset of skills from $\mathcal{M}$. Let S_g be the skill set of group g. We assume that for each shift there is a group of agents that have the skills to work that shift. Hence, for notational convenience we can denote the skill set of each shift

by f_k, i.e., the index of the corresponding agent group for shift k. In order to meet the service-level constraints, we suppose that for every agent group there is a set of workable shifts such that for some agent configuration the requirements are met. In this context, a shift k is workable if there is a group g such that $f_k = g$, and agents that work in shift k can work in all groups g' that satisfy $S_{f_k} \subseteq S_{g'}$.

The two steps of the scheduling algorithm are discussed in the next two sections.

9.3.1 Staffing levels

The purpose of the first part of the algorithm is to determine a nearly optimal staffing configuration. With respect to the optimization of staffing levels we refer to Chapter 8. This algorithm requires the costs of staffing an agent in each of the different groups. We calculate the cost of scheduling s_g agents in group g, as denoted in Chapter 8 by $K^g(s_g)$, according to the following formula

$$K^g(s_g) := s_g \sum_{k:f_k=g} \frac{c_k}{a_k e}, \quad \text{with } e \text{ the unit-vector.}$$

It is the average cost of the possible shifts the agents from the group can work, normalized by the shift lengths.

9.3.2 Shift scheduling

This section describes the second step of the two-step algorithm. We answer the question of how to determine the optimal number of shifts for each type. We also answer the question of how to allocate agents to the different agent groups in each period.

These questions are answered by means of an integer programming model. The objective of the integer programming model is to minimize personnel costs while meeting the staffing requirements for each group in each period. The main feature of this model is that agents can work in different groups during the same shift. The skill set of the shift determines if an agent with a specific type of shift is allowed to work in a certain agent group. An agent with skill set X is allowed to work in a group with skills X' if $X' \subseteq X$. The input of the model consists of the required number of agents $s_{t,g}$ from step 1. Hence, information about the arrival streams, control policy, and service time distributions is not used in this step.

Example

We introduce the integer programming model by means of an example. Consider a call center with three skills $\mathcal{K} = \{1, 2, 3\}$ and six agent groups $S_1 = \{1\}, S_2 = \{2\}, S_3 = \{3\}, S_4 = \{1, 2\}, S_5 = \{2, 3\}$, and $S_6 = \{1, 2, 3\}$. The model is depicted in Figure 9.1, showing the agent groups and the arrival streams.

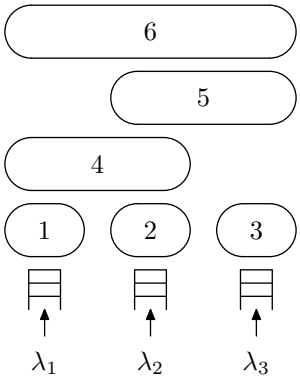


Figure 9.1: Example of a three-skill call center

We introduce an integer programming model to determine the cheapest set of shifts such that the requirements on the agent numbers are met. To give more insight in the model, we assume that the optimal values x_k and the number of agents working shift k, are known. Then for each time interval t the assignment of the available agent numbers x_k to the agent groups can be modeled as a linear assignment problem. This is depicted as a graph in Figure 9.2. The nodes on the left represent the scheduled number of agents for each skill set, which is determined by the variables x_k. Note that the number of scheduled agents with skills $\mathcal{M}'$ is equal to

$$\sum_{k:S_g=\mathcal{M}',a_{k,t}>0} x_k,$$

with $g \equiv f_k$ and $\mathcal{M}' \subseteq \mathcal{M}$. The nodes on the right denote the required number of agents per agent group. The agents available on the left side need to be assigned to the agent groups on the right. A feasible solution of the linear assignment problem results in a feasible assignment of the scheduled

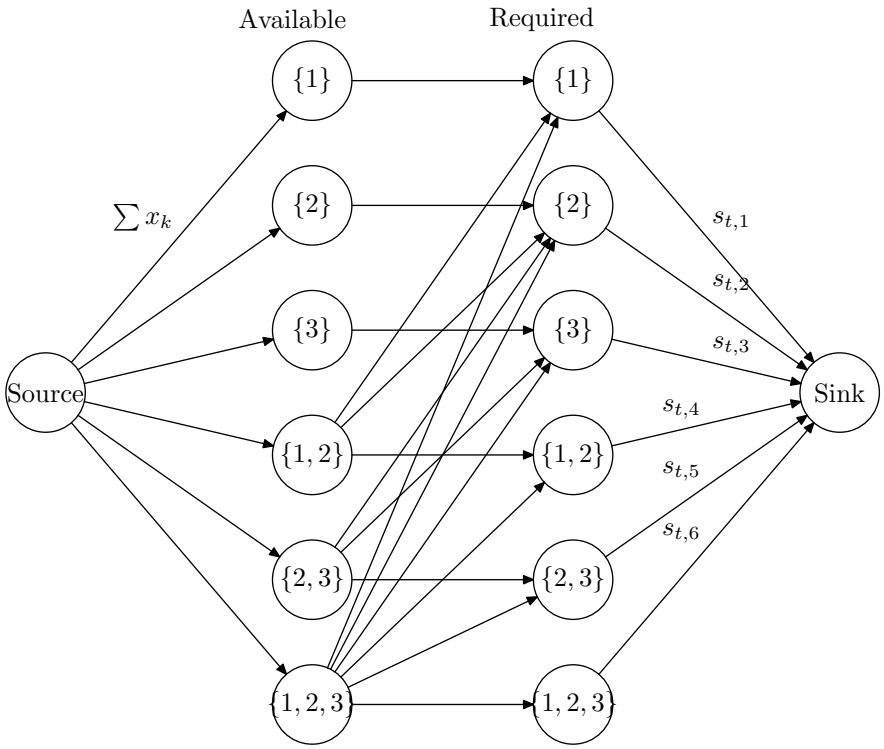


Figure 9.2: Linear assignment problem

agents to the different agent groups. However, the assignment of available agents from Figure 9.2 to the agent groups is not explicitly modeled in the integer program because a reduction of decision variables is possible. The reduction is obtained by introducing dummy variables $y_{g',g,t}$ for each group g', with g such that $S_g \subseteq S_{g'}$. The variable $y_{g',g,t}$ denotes the number of agents that are removed from group g' and work in group g that have fewer skills at time t. Note that any subset $\mathcal{X}'$ of $\mathcal{X}$ can be obtained by removing elements successively. Therefore, the dummy variables $y_{g',g,t}$ make all feasible assignments possible. For example, an agent from group 6 can operate as a specialist in group 1 by setting the two dummy variables $y_{6,4,t}$ and $y_{4,1,t}$ to 1. We can depict the dummy variables by arcs between groups that have one skill less as shown in Figure 9.3. The introduction of the dummy variables leads to a significant reduction in decision variables. Suppose that we

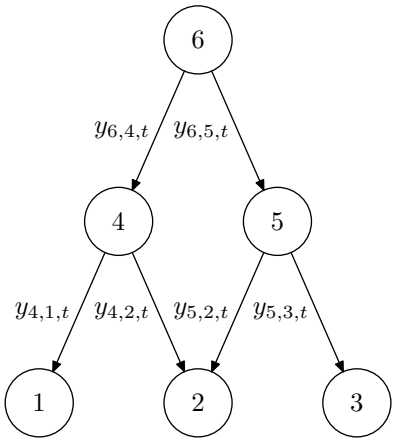


Figure 9.3: Simplified assignment model

have a call center with groups having all possible combinations of skills. The linear assignment model has $\sum_{k=1}^{M} \binom{M}{k}(M^k - 1) = 3^M - 2^M$ variables. The simplified model has $\sum_{k=2}^{M} \binom{M}{k} k = M(2^{M-1} - 1)$ variables when $y_{g',g,t}$ has the additional constraint $|S_g| = |S_{g'}| - 1$. If not all combinations of skills are represented by a group, then the number of decision variables can be reduced further, as we will explain in the following section.

We remark that any subset $\mathcal{X}'$ of $\mathcal{X}$ can be obtained by removing elements successively only if all types of agent groups are present. However, in practice call centers often have a limited number of groups, which we also allow in our model formulation. Hence, we choose the dummy variables more carefully in the integer programming model, that we formulate in the next section.

Model

For the model, the necessary dummy variables are determined as follows. We consider each period t, with $t \in \mathcal{T}$. For notational convenience we define the set $\mathcal{G}_t$, containing a subset of the agent group indices that are required at time t. Group g is included in $\mathcal{G}_t$ if

- $s_{t,g} > 0$, or
- $a_{k,t} > 0$ for some $k \in \mathcal{K}$ and $f_k = g$.

Thus, it contains the indices of agent groups with a positive number of required agents or a potential positive number of scheduled agents (by having a shift with the same skills). Next, we define two sets of decision variables: $\mathcal{I}_{t,g}$ and $\mathcal{J}_{t,g}$. Set $\mathcal{I}_{t,g}$ contains the decision variables associated with agents moving from higher level groups to agent group g and set $\mathcal{J}_{t,g}$ contains the decision variables associated with agents moving from group g to lower level groups. Variable $g' \in \mathcal{I}_{t,g}$ is included in the model if

- $g', g \in \mathcal{G}_t$,
- $S_g \subset S_{g'}$, and
- there exists no $g^* \in \mathcal{G}_t$ such that $S_{g'} \supset S_{g^*} \supset S_g$,

and variable $g' \in \mathcal{J}_{t,g}$ is included if

- $g', g \in \mathcal{G}_t$,
- $S_{g'} \subset S_g$, and
- there exists no $g^* \in \mathcal{G}_t$ such that $S_{g'} \subset S_{g^*} \subset S_g$.

Note that we require that no strict subset S_{g^*} exists between $S_{g'}$ and S_g. By using this notation we can describe the integer programming model as

$$\min \sum c_k x_k$$

subject to

$$\sum_{k \in \mathcal{K}: f_k = g} a_{k,t} x_k + \sum_{g' \in \mathcal{I}_{t,g}} y_{g',g,t} - \sum_{g' \in \mathcal{J}_{t,g}} y_{g,g',t} \geq s_{t,g}, \qquad \forall t \in \mathcal{T}, \forall g \in \mathcal{G}_t,$$

$$x_k, y_{g',g,t} \geq 0 \text{ and integer}, \qquad \forall k \in \mathcal{K}, \forall t \in \mathcal{T}, \forall g, g' \in \mathcal{G}_t.$$

When having obtained a solution (x, y), the schedule is composed, specifying for each shift the agent groups that the agent works in, during each period.

This is done according to the following algorithm.

```
SHIFT COMPOSITION()
 1  Choose k such that x_k ≥ 1. Set agent group g = f_k.
 2  For each period t with a_{k,t} = 1:
 3      Initialize ḡ ← g.
 4      Repeat:
 5          If variable y_{g,g',t} exists and y_{g,g',t} > 0 for some g'
 6          ḡ ← g' and y_{g,g',t} ← y_{g,g',t} − 1
 7          else stop and assign the shift to group ḡ at time t.
 8  End for
 9  Decrease x_k by one and go to line 1 unless x_k = 0 for all k.
```

As we will show in Section 9.4.2, the problem in case of two or three skills is numerically tractable. According to the literature, we can expect that this also holds for cases with a much larger number of different skills, and many different types of shifts. Literature shows that set-covering problems are relatively easy to solve. There are a huge number of papers available on crew scheduling on trains and airplanes. In particular, we would like to mention the shift-scheduling problems, in which tasks are paired to shifts. Studies show that problems of over thirty thousand tasks are solved within reasonable time, e.g., hours, with shifts including breaks and many other features. In these problems, each task corresponds to a constraint, similar to a staffing level in our problem. The largest problems are solved close to optimality by using column generation in conjunction with a Lagrange relaxation, see for example Caprara, Fischetti, and Toth (1999), which also is applicable to our integer programming problem.

9.3.3 Global service-level constraints

The purpose of this section is to show that the algorithm from Section 9.3.2 can be extended to handle constraints on the average service level during the day. To this end, we describe a local-search method for determining staffing levels such that this type of constraint is satisfied. We illustrate that the extension behaves well in Section 9.4.3, by means of a numerical experiment.

First we give two observations underlying the heuristic:

- Consider the integer programming model for the multi-skill call center. According to the model, each row corresponds to both a period and

an agent group, which is denoted by the right-hand side of the model. It is well known from duality theory that a dual variable indicates the relation between the right-hand-side and the value of the objective function. Thus, agents that contribute to the staffing level of a group with a high dual value, in a certain period, have a larger contribution to the objective value.

- With regard to service levels, not all adjustments to the staffing levels are equally preferred. For example, if we decrease $s_{t,g}$ by 1 and increase $s_{t',g'}$ by 1, we prefer to choose t, g, t', and g' such that the resulting overall service level is highest.

By combining both observations we construct the following algorithm:

Step 1: Solve the integer programming model from Section 9.3.2. This gives the initial solution.

Step 2: Select the m rows with the highest dual values and the m rows with the lowest dual values. Pick one row from each selection, which denotes a movement of an agent from one group to another and choose both rows in such a way that the decrease in service level is minimal. Repeat this step as long as the changes are significant. Otherwise, go to step 3.

Step 3: If the overall service level exceeds the minimum requirement, select the m rows with the highest dual values and decrease one of these group sizes such that the decrease in service level is minimal. If the overall service level is not sufficient, select the m rows with the lowest dual values and increase the right-hand-size coefficient of one of the rows such that the service level is maximized. Repeat this step until the average service level is just above the bound. Otherwise stop or, if significant changes are made, go to step 2.

To measure the relative differences of the service level, we used the algorithms from the literature for multi-skill blocking systems, as mentioned in Chapter 8. The experiments show, of which one is provided in Section 9.4.3, that it reduces the calculation speed of this algorithm significantly and yields good results.

9.4 Numerical experiments

This section consists of two parts. The first part provides a numerical example of the method discussed in this chapter. The second part discusses a small realistic example that combines the method of this chapter with the method from Chapter 8.

9.4.1 Scheduling algorithm

In this section we give an example of a shift-scheduling problem for a call center with three skills and with a high variability in groups sizes, $S_1 = \{1\}$, $S_2 = \{2\}$, $S_3 = \{3\}$, $S_4 = \{1, 2\}$, $S_5 = \{2, 3\}$, $S_6 = \{1, 3\}$, and $S_7 = \{1, 2, 3\}$. The day consists of fourteen periods, indexed from 1 to 14. We consider a limited number of shift types: Generalists work in shifts of length 6, have a cost of 6 and start in intervals 1–9. Cross-trained agents work in shifts with: skills 1 and 3, a length of 5, a cost of 5 and start in intervals 1–10, skills 2 and 3, a length of 6, a cost of 5.5 and start in intervals 1–9, and skills 1 and 2, a length of 5, a cost of 4.5 and start in intervals 1–10. Specialists of type 3 work in shifts of length 6 that start in intervals 1–6 and have a cost of 5. Specialists of type 2 work in shifts with a length of 6 that start in intervals 7–9, having a cost of 5, and they also work in shifts with a length of 5 starting in 1–3, and a cost of 4. Specialists of type 1 start in intervals 4–10 with shifts of length 5, and cost 4.

We composed staffing vectors manually for a day consisting of fourteen intervals, see Table 9.1. The diversity of the agent types in each vector is higher than in reality; the number of specialists is relatively low and the distribution of the agents among the groups is diverse over the different intervals. The optimal solution is displayed in Appendix C. The total number of shifts is 35, and the total number of idle periods is 16. The solution requires 8 generalists, 18 agents with two skills, and 9 specialists, including 16 shifts of length 6 and remainder of length 5.

As a comparison, we calculate the total number of agents in each interval and consider a single-skill call center. We solve the problem from Section 9.2.1 by using the online tool that is available at `http://www.math.vu.nl/~sapot/software/shift-scheduling`, which includes column generation. The optimal solution also requires 35 shifts, including 10 shifts of length 6. Thus, the total idle time is 6 periods less than the number that the multi-skill solution provides. Note that 6 is only a lower bound for the total

$g\backslash t$	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	1	2	3	2	2	3	2	1	1	2	2	1	1	1
2	1	2	3	2	2	3	2	1	1	2	2	1	1	1
3	1	2	2	2	2	1	2	1	1	1	0	1	1	1
4	1	2	2	3	2	3	2	3	3	1	0	1	1	1
5	1	1	2	3	2	3	2	3	3	2	2	2	1	1
6	1	1	2	2	2	2	2	3	3	2	2	2	1	1
7	1	3	2	2	2	2	4	3	3	2	2	1	2	1

Table 9.1: Required group sizes, $s_{t,g}$

$m\backslash t$	1	2	3	4	5	6	7	8	9	10
1	0.37	1.15	1.43	1.47	1.33	1.55	1.47	1.48	1.37	1.02
2	0.80	1.68	2.03	2.28	2.22	2.38	2.27	2.15	2.13	1.60
$m\backslash t$	11	12	13	14						
1	0.72	0.68	0.60	0.37						
2	1.35	1.32	0.98	0.82						

Table 9.2: Arrival rates, $\lambda_{m,t}$

idle time, because in the multi-skill case not all shift types have a length of 5 and 6; some have a length of 5, and others a length of 6.

9.4.2 Case study

This study is based on the statistics of a Dutch call center. The arrival rates are given in Table 9.2. The service rates are $\mu_{1,1} = 0.186, \mu_{2,1} = 0.577, \mu_{3,1} = 0.169$, and $\mu_{3,2} = 0.526$. Three agent groups are distinguished, having indices 1, 2, and 3. We consider shifts with a length of 5 and 6 hours. The costs of a five-hour shift is 5 for generalists, 4.5 for specialists of type 1, and 4 for specialists of type 2. The costs of a six-hour shift is 6, 5.5, 5 for generalists, specialists of type 1, and specialists of type 2, respectively.

We apply the two-step method from Section 9.3. The optimal solutions of the mathematical programming models are determined by means of an integer linear programming solver. We used SA-OPT[1] that is written by the author of this thesis. The result of step 1 is presented in Table 9.3.

[1]See `http://www.math.vu.nl/~sapot/software/sa-opt` for technical details

$g\backslash t$	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	2	4	6	5	5	6	7	8	7	5	5	3	3	2
2	2	4	6	5	5	6	5	5	4	4	5	3	3	2
3	2	5	4	6	5	5	4	3	4	3	0	3	2	2

Table 9.3: Required group sizes, $s_{t,g}$

The table shows the required number of agents for each period and agent group. The optimal set of shifts according to the model from Section 9.3.2 is presented in the second table of Appendix C with an objective of 167. Each row represents a shift, specifying the group that the agents works in, for each interval. The columns represent the different periods. The solution consists of 10 shifts with skills 1 and 2, 14 shifts with skill 1, and 11 shifts with skill 2. The value 3 indicates that the agents work in agent group 3, i.e., the group of generalists. The value 0 denotes idleness, meaning that conditions concerning the service level are already satisfied such that the employee is redundant in that period.

We note that in the optimal schedule a generalist sometimes works as a specialist. This is beneficial because specialists have a higher service rate. Furthermore, we see that an agent is sometimes idle during a shift. These idle periods can be used for serving other contact channels (such as emails and faxes) and training, without compromising the service level.

If we relax the problem and impose a constraint only on the average service level during the day, we can apply the algorithm from Section 9.3.3. The objective value decreases from 167 to 161, after five adjustments to the group sizes, see Section 9.4.3 for a description of the iterations. To check the optimality, we made a comparison to a call center with one skill. One skill simplifies the calculations, because now we can determine the optimal schedule according to the programming model described in Section 9.2.2 (and using the Erlang-C formula). The experiment showed that both algorithms resulted in schedules with the same objective value. Thus, the heuristic yielded optimal results in this case.

To check the optimality of the methods from Section 9.3.1 and 9.3.2, we evaluated the results from Appendix C. Two methods were considered for obtaining lower bounds for the objective function. First we extended the integer programming model from Section 9.2.2 to a multi-skill call center. Unfortunately, the number of decision variables turned out to be very

large (hundreds of thousands) and, given the fact that all variables must be integer, we were not able to obtain a feasible solution. However, without satisfying the integrality requirement, we did succeed in finding an optimal solution, yielding a lower bound of 150. This result was not satisfactory because the gap between 150 and 167 is relatively large. Hence, we considered a second approach for obtaining a lower bound. We determined a lower bound for the cost of an agent working during one time interval in a certain group. This can easily be derived from the costs of the shifts, yielding 0.8, 0.9, and 1.0 for specialists of type 1, type 2, and generalists, respectively. Next, for each period we calculated the cheapest agent configuration. Since there are only three agent groups this is doable by enumerating all possible configurations and executing simulations. The lower bound of the total cost for the whole day is calculated by multiplying the group sizes by the costs and by summing over the intervals, yielding 154.3. Nevertheless, the optimal solution can still be higher than 154.3 because it is likely that the optimal set of shifts exceeds the staffing levels at certain periods, resulting in idle times as we saw in Appendix C. We realized that we can calculate a tighter lower bound by determining the idle time that is minimally required. To achieve this, we calculated the minimum number of required agents for each time interval (by summation of the number of specialists and generalists) and considering a single-skill call center. We solved[2] the model from Section 9.2.1 and concluded that the minimum idle time is ten periods. Then the lower bound becomes 154.3+0.8*10 = 162.3. This shows that the solution from Appendix C is less than 3% from optimality.

9.4.3 Experiment with global constraints

We apply the algorithm from Section 9.3.3 to the example of Section 9.4.2. Parameter m is set to 5. The initial solution s is determined by Table 9.3, yielding a service level of 81% and an objective value of 167. The corresponding values of the dual variables vary between 0.0 and 2.0. The iterations are described next:

1. $s_{2,3} \leftarrow s_{2,3} - 1$ and $s_{5,1} \leftarrow s_{5,1} + 1$. Afterwards, the service level becomes 81% and the objective value is 166. The dual costs of the adjusted staffing levels are 1.9 and 0.0, respectively , and all dual costs varied between 0.0 and 2.0.

[2] An online tool is available at `http://www.math.vu.nl/~sapot/software/shift-scheduling`

2. $s_{3,2} \leftarrow s_{3,2} - 1$ and $s_{10,3} \leftarrow s_{10,3} + 1$, the service level becomes 81% and the objective value becomes 165. The dual costs of the selected staffing levels are 1.8 and 0.0, and all dual costs varied between 0.0 and 1.8.

3. $s_{12,3} \leftarrow s_{12,3} - 1$ and $s_{6,3} \leftarrow s_{6,3} + 1$, the service level is 81% and the objective value is 164. The dual costs of the selected staffing levels are 1.6 and 0.0, and all costs varied between 0.0 and 1.7. The staffing level associated with the dual value of 1.7 is not selected because the service level would decrease too much, compared to the other possible adjustments.

4. $s_{4,3} \leftarrow s_{4,3} - 1$ and $s_{10,1} \leftarrow s_{10,1} + 1$, a service level of 80% and the objective values becomes 162. The duals range between 0.0 and 1.5 and the adjusted staffing levels have dual values of 1.3 and 0.0, respectively.

5. $s_{14,3} \leftarrow s_{14,3} - 1$ and $s_{1,1} \leftarrow s_{1,1} + 1$, with duals ranging from 0.0 up to 2.0, and values of 1.9 and 0.2 are associated with the selected groups and time intervals.

After this iteration no better result was found. The objective value of the solution after these five iterations is 161 and the service level is 80% on average during the day.

9.4.4 Discussions

The main algorithm that we introduced consists of two steps. A possible drawback of determining staffing levels and generating shifts separately, in two steps, is suboptimality. This is to some extend avoided by the condition from Section 9.3.2, i.e., there should be at least one shift available for each group. However, if there are many different agent groups with only a few number of skills and all shifts require only a small number of skills, one should be careful. Consider for example a call center with four skills: A, B, C, and D, and four agent groups with skill sets $S_1 = \{1, 2\}$, $S_2 = \{3, 4\}$, $S_3 = \{1, 3\}$, $S_4 = \{2, 4\}$, and $\mathcal{T} = \{1, 2\}$. We define $S = (S_1, S_2, S_3, S_4)$. If the staffing algorithm produces for the first interval of the day $S = (1, 1, 0, 0)$ and for the second interval $S = (0, 0, 1, 1)$, the shift-scheduling algorithm would consist of four shifts instead of two, such that agents are idle for 50% of the time. This could be avoided by choosing the agent groups more carefully, or by improving the staffing vectors by a local-search algorithm if the idle time of

the shifts is high. In our opinion, this is an interesting subject for future research. We remark that a lower bound for the amount of idle time can be determined according to the method given in Section 9.4.2.

We expect that the inaccuracy from the previous example will be small in practice. First of all, in many call centers the number of different skills is limited, which makes the inaccuracy less likely. Secondly, our experience is that if there are at least a few agent groups with more than two skills, the results of the algorithm become nearly optimal. The reason is that solutions obtained by the algorithm prescribe, in realistic cases, the usage of relatively many specialists, since specialists are cheaper and work faster, and solutions often only require relatively few cross-trained agents with two skills, and hardly any agent with more than two skills. By including a few agent groups with more than two skills, the time that agents are idle is expected to be low. A disadvantage is that solutions could require agents with additional skills compared to an optimal solution. This is still suboptimal if agents with more skills are significantly more expensive. However, call centers often prefer a sufficient amount of flexibility of agents in case the actual workload deviates from the predictions, such that agents can be rescheduled. Then it is desired to have agents available with additional skills. As a result, call centers often have a sufficient number of agents with more than two skills.

Although it is out of the scope of this chapter, we remark that suboptimality can be significant in phase 4 of the labor allocation process. This phase concerns the assignment of shifts to employees. Suboptimality can occur if insufficient employees are available to satisfy the requirements for a type of shift, requiring a specific set of skills. There are several ways to avoid this. For example, by only creating agent groups with skill sets that occur among the agents. In addition, the staffing algorithm can be extended by adding constraints on the group sizes or on the sums of several group sizes. Afterwards, by studying the results from the shift-scheduling step and changing the staffing levels, it is likely that further improvements are possible.

9.5 Extensions

In this section we discuss a number of extensions. These additional features extend the model to more specific scenarios, and make the algorithm more useful in practice. The first extension explains how to reserve resources for

emails and faxes. The second extension relaxes the service-level conditions to decrease costs without much loss in performance. The third extension discusses the inclusion of bounds on the sets of shifts in the models. This may be desired when assigning shifts or rosters to employees, such that the availability of employees can be taken into account more easily.

We illustrate the extensions by applying them to the single-skill shift-scheduling model. However, they can also readily be applied to the multi-skill integer programming models from Section 9.3.2.

Emails and faxes

In many call centers nowadays, emails and faxes represent a substantial part of all traffic, next to the telephone calls. This has a positive impact on the productivity of these 'contact centers'. The explanation is as follows. Since emails and faxes have lower service-level requirements than calls, managers can let emails and faxes wait at times where many telephone calls are in the system or are expected to arrive. Hence, they are only handled in periods that are not too busy. As a result the workload is more constant over the day. This flexibility of scheduling emails and faxes improves working schedules because idle times disappear.

We define q as the desired number of hours spent on handling emails and faxes. The decision variable $\hat{x}_t$ denotes the number of agents that work on emails and faxes at time t. The integer programming model then becomes

$$\min \sum_{k \in \mathcal{K}} c_k x_k$$

subject to

$$\sum_{k \in \mathcal{K}} a_{k,t} x_k - \hat{x}_t \geq s_t, \qquad \forall t \in \mathcal{T},$$
$$\sum_{t \in \mathcal{T}} \hat{x}_t = q,$$

$$x_k, \hat{x}_t \geq 0 \text{ and integer}, \quad \forall k \in \mathcal{K}, \forall t \in \mathcal{T}.$$

Penalties

By using the results of Keith (1979), the different models from this chapter can be refined to weaken the constraints on the minimum personnel requirements.

Let the parameter $\bar{p}_t$ denote the cost per agent exceeding s_t during time interval t, and let $\tilde{p}_t$ denote the cost per agent falling short of s_t during time interval t. Define the decision variable $\bar{x}_{k,t}$ to denote the redundant number of agents in shift k that work during time interval t, and $\tilde{x}_{k,t}$ to denote the shortage of agents in shift k that work during time interval t. The integer programming model then becomes

$$\min \sum_{k \in \mathcal{K}} c_k x_k + \sum_{t \in \mathcal{T}} (\bar{p}_t \bar{x}_t + \tilde{p}_t \tilde{x}_t)$$

subject to

$$\sum_{k \in \mathcal{K}} a_{k,t} x_k - \bar{x}_t + \tilde{x}_t = s_t, \qquad \forall t \in \mathcal{T},$$

$$x_k, \bar{x}_t, \tilde{x}_t \geq 0 \text{ and integer}, \qquad \forall k \in \mathcal{K}, \forall t \in \mathcal{T}.$$

Lower and upper bounds

To take the availability of agents into account, a lower and upper bound can be specified for each shift. In case of N bounds, this is done as follows, by means of N constraints. Let S_i denote the set of shifts, by means of indices, to which constraint i is applicable, $i \in \{1, 2, \ldots, N\}$ and $S_i \subset \mathcal{K}$. We add a constraint to impose a lower and upper bound on the total number of occurrences of each shift type. Let parameter l_i be the lower bound on the number of shifts from set S_i, and u_i the upper bound on the number of shifts

from set S_i. The integer programming model is then given by

$$\min \sum_{k \in \mathcal{K}} c_k x_k$$

subject to

$$\begin{aligned} &\sum_{k \in \mathcal{K}} a_{k,t} x_k \geq s_t, && \forall t \in \mathcal{T}, \\ &l_i \leq \sum_{m \in S_i} x_m \leq u_i, && \forall i \in \{1, \ldots, N\}, \\ &x_k \geq 0 \text{ and integer}, && \forall k \in \mathcal{K}. \end{aligned}$$

9.6 Concluding remarks

The contribution of this chapter are methods for shift scheduling in multi-skill call centers, consisting of the estimation of staffing levels and, the corresponding optimization of the set of shifts. These are the first methods in the literature that are computationally tractable, such that shifts can be generated for multi-skill call centers. Although our experiments deal with two and three skills, computations are still tractable for call centers with more skills; we experienced and can reason that the computation times for large call centers are in the order of minutes, and is favored above the optimization procedures from Section 9.3.1, which are of logarithmic order in the size. Another strength is that the methods are easy to implement.

As a possible extension, it might be necessary or beneficial to perform the two steps of the algorithm several times and iteratively. This is for example desirable if scheduled agents become ill and agents have to be rescheduled, or if workload predictions of a certain job type change. For that reason, it is likely that call center managers will prefer fast methods for each separate phase of workforce management, such that they can iterate between the four phases within a short time period.

There are different avenues for future research. It is straightforward to use the model from Section 9.3.2 to multi-skill rostering, i.e., combining daily shifts to weekly rosters. The main difference is that the rows represent

the shifts, instead of required group sizes, and each column represents the weekly schedule of an agent, instead of shifts. Then the schedules can be assigned to the available employees afterwards. To handle the large number of possible schedules, column generation (a well-known method from linear programming) can be used. These problems are numerically tractable and have short computation times. This even has the potential to solve phases 3 and 4 at once. Another direction for future research is to analyze the heuristic from Section 9.3.3 and to do more experiments.

A promising method is the method of Cezik and L'Ecuyer (2005). The advantage of their method is that it takes the transient behavior into account and that it is theoretically possible to solve phases 2 and 3 simultaneously. It would be interesting to replace the method from Section 9.3.1 by theirs and to combine their method with the integer problem from Section 9.3.2. This would be beneficial if the efficiency of their algorithm can be increased because solutions are not numerically tractable.

Chapter 10

Conclusions

In this section we look back on the content of this thesis, make our final statements, and give directions for future research.

Summary

This thesis treated several subjects on routing and scheduling in multi-skill contact centers. It started with an overview of the literature and a description of the model. Next, standard techniques from the literature were considered, which showed that numerical calculations are hardly tractable due to the high complexity of these systems. This motivated us to consider approximation techniques. The results of this study were discussed in the thereupon following chapters. They cover methods that can help call centers to:

- determine routing policies for the assignment of jobs to agents,
- update the routing policies by using real-time information,
- approximate performance measures in order to improve staffing configurations,
- optimize staffing levels to minimize costs, and
- compose schedules of working shifts such that idle time is minimized.

In particular, these methods are useful for the planning of agents, and to adapt and optimize routing policies and working schedules of agents to unpredicted events during operations.

Conclusions

In this section we mention some general conclusions that one can draw from the content of this thesis. For detailed conclusions about one of the previous chapters we refer to the section Concluding Remarks of the chapter. The general conclusions are that:

- accurate computations on planning and routing problems are hardly tractable for call centers of a realistic size such that approximations become important,
- researchers in mathematics can contribute significantly to the optimization of business processes in call centers,
- the gain by mathematical optimization can be substantial, which is important for economical reasons, and
- call centers are an important subject for future research because many open questions remained unanswered. This is addressed more extensively in the next section.

We note that these conclusions follow directly from the results of the previous chapters.

Future directions

During the last decades there has been a trend in the direction of 'virtual contact centers'. The rapid developments in tele-communications (e.g., fast internet connections and protocols) and in computer technology (e.g., powerful automatic call distributors) make it possible to bring employees from different locations together, explaining the word 'virtual'. Telephone is the traditional way of communication in call centers. Currently other media such as email and fax are also very popular. We expect that the number of virtual contact centers will increase. The future expectation is that agents log on to the system from their home, resulting in higher flexibility of both parties. Also more types of media will be supported by the contact centers, for example chat and self-service media such as video. An appropriate synonym for virtual contact centers is "single-point-of-contact help desk", mentioned in Wallace and Whitt (2005).

Our impression of the directions for future work is that there are still great challenges for mathematicians. An interesting subject for future research is the integration of routing and planning in the models and algorithms. By considering problems of a larger scope, results can become even more useful for the industry.

The new types of media will increase the variation in skills and service-level constraints. This yields even more complex mathematical problems. Other business models, such as outsourcing and offshoring, will probably contribute to the variety.

In our opinion, the online optimization of routing policies, as discussed in Chapter 6, deserves more attention. We expect that this is an important and relevant subject for the industry.

Next, models from the literature often use many assumptions and require approximations, concerning for example the arrival-time and service-time distributions. The impact of assumptions and the estimates of model parameters are complicated to analyze. It is in our opinion important to validate the models in realistic situations. For example, how effective are call routing policies if the workload deviates from the predictions. In our opinion, research about model validation and robustness is often lacking, but important for the industry.

Appendix A

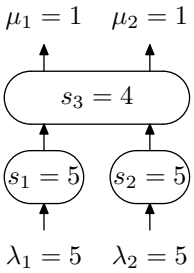
Numerical Comparison of Approximation Methods

The literature provides different methods to approximate blocking probabilities of overflow blocking systems, as discussed in Chapter 7. In this appendix, we present the numerical results of these methods that result from the application to a total of eighteen different blocking systems. Each system is depicted in a figure, which completely specifies the parameters. We refer to Section 7.2 for an explanation.

The blocking probabilities are calculated by seven different methods: exact calculation (EXACT), simulation (SIM), Exponential Decomposition (ED), HyperExponential Decomposition (HED), the Equivalent Random Method (ERM), Hayward-Fredericks method (HF) and the Interrupted Poisson Process method (IPP), as described in Chapter 7. The results are given below each of the figures. We remark that the results of the methods EXACT and ERM are missing in certain cases. This denotes either that calculations are intractable or that the method requires a more restricted type of overflow routing policy, respectively.

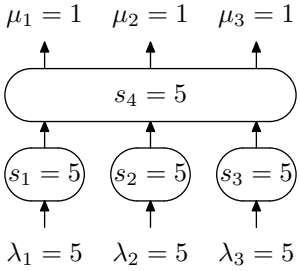
In order to make a fair comparison between the different approximation methods, a large variety of routing policies, arrival rates, group sizes and service rates are considered. The parameters are chosen in such a way that the service level would be realistic in case of a delay system.

Instance 1



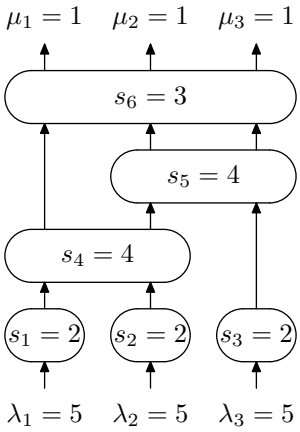
	skills		
method	1	2	
EXACT	0.0792	0.0792	0.0792
SIM	0.0792	0.0791	0.0791
ED	0.0539	0.0539	0.0539
HED	0.0791	0.0791	0.0791
ERM	0.0800	0.0800	0.0800
HF	0.0781	0.0781	0.0781
IPP	0.0766	0.0766	0.0766

Instance 2



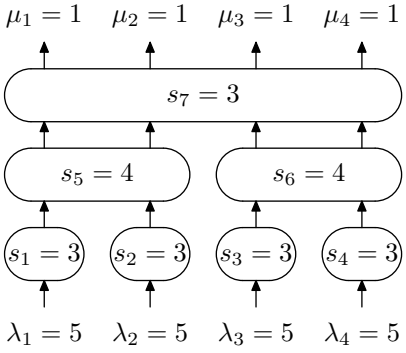
	skills			
method	1	2	3	
SIM	0.0836	0.0837	0.0839	0.0837
ED	0.0636	0.0636	0.0636	0.0636
HED	0.0833	0.0833	0.0833	0.0833
ERM	0.0876	0.0876	0.0876	0.0876
HF	0.0825	0.0825	0.0825	0.0825
IPP	0.0797	0.0797	0.0797	0.0797

Instance 3



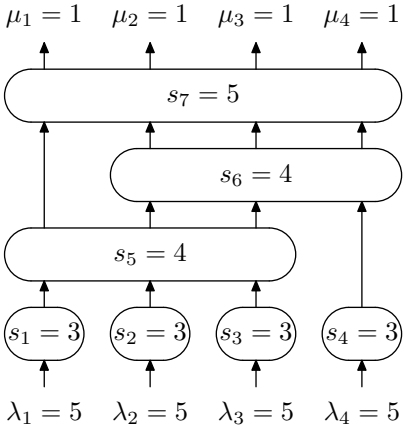
	skills			
method	1	2	3	
SIM	0.1737	0.0971	0.1527	0.1412
ED	0.1509	0.0615	0.1195	0.1106
HED	0.1744	0.0933	0.1586	0.1421
HF	0.1903	0.0852	0.1605	0.1453
IPP	0.1702	0.0734	0.1394	0.1277

Instance 4



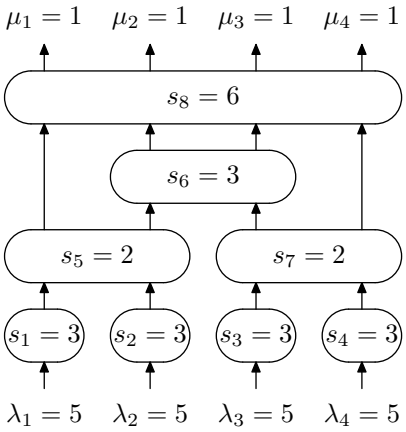
	skills				
method	1	2	3	4	
SIM	0.1368	0.1365	0.1363	0.1368	0.1366
ED	0.1092	0.1092	0.1092	0.1092	0.1092
HED	0.1383	0.1383	0.1383	0.1383	0.1383
ERM	0.1397	0.1397	0.1397	0.1397	0.1397
HF	0.1389	0.1389	0.1389	0.1389	0.1389
IPP	0.1264	0.1264	0.1264	0.1264	0.1264

Instance 5



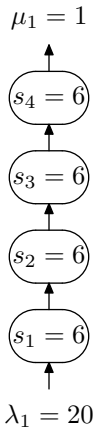
	skills				
method	1	2	3	4	
SIM	0.0915	0.0612	0.0621	0.0837	0.0746
ED	0.0620	0.0278	0.0278	0.0486	0.0415
HED	0.0889	0.0633	0.0633	0.0918	0.0768
HF	0.1076	0.0540	0.0540	0.0915	0.0768
IPP	0.0749	0.0352	0.0352	0.0602	0.0514

Instance 6



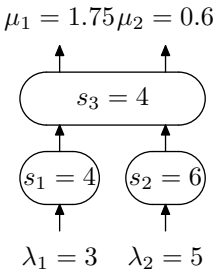
	skills				
method	1	2	3	4	
SIM	0.1033	0.0609	0.0615	0.1028	0.0821
ED	0.0752	0.0314	0.0314	0.0752	0.0533
HED	0.1048	0.0613	0.0613	0.1048	0.0830
HF	0.1156	0.0560	0.0560	0.1156	0.0858
IPP	0.0906	0.0408	0.0408	0.0906	0.0657

Instance 7



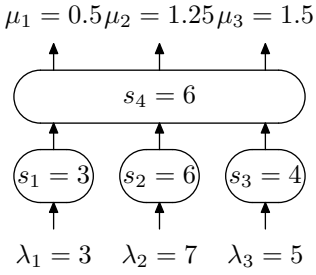
method	skills 1	
SIM	0.0661	0.0661
ED	0.0211	0.0211
HED	0.0660	0.0660
ERM	0.0661	0.0661
HF	0.0630	0.0630
IPP	0.0658	0.0658

Instance 8



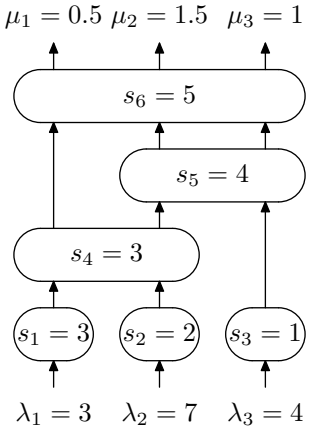
method	skills 1	2	
EXACT	0.0201	0.1409	0.0956
SIM	0.0201	0.1410	0.0957
ED	0.0175	0.1065	0.0731
HED	0.0201	0.1408	0.0955
HF	0.0225	0.1371	0.0941
IPP	0.0227	0.1385	0.0951

Instance 9



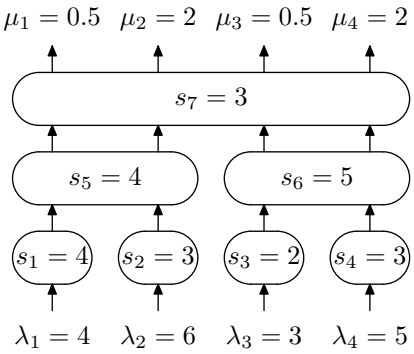
	skills			
method	1	2	3	
SIM	0.1587	0.0777	0.0726	0.0922
ED	0.1426	0.0571	0.0586	0.0747
HED	0.1584	0.0774	0.0728	0.0921
HF	0.1733	0.0694	0.0712	0.0908
IPP	0.1669	0.0669	0.0686	0.0875

Instance 10



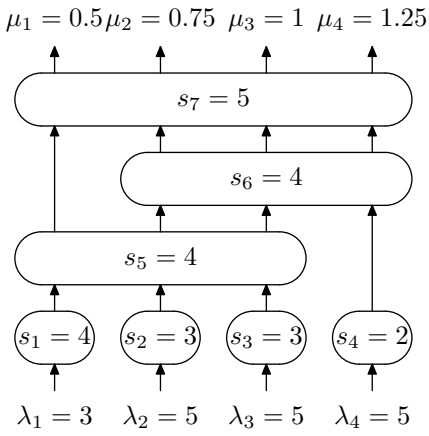
	skills			
method	1	2	3	
SIM	0.1098	0.0741	0.1080	0.0914
ED	0.0817	0.0370	0.0726	0.0567
HED	0.1061	0.0702	0.1146	0.0906
HF	0.1234	0.0607	0.1164	0.0901
IPP	0.0989	0.0470	0.0905	0.0706

Instance 11



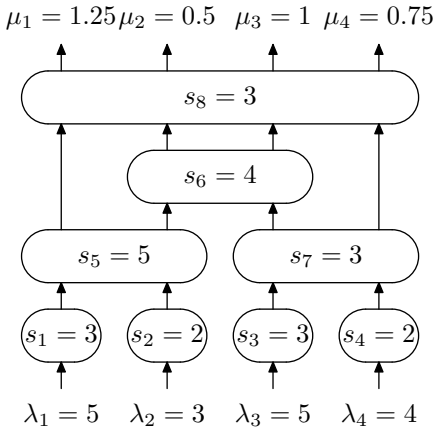
	skills				
method	1	2	3	4	
SIM	0.1481	0.0936	0.1178	0.0549	0.0990
ED	0.1142	0.0688	0.0922	0.0361	0.0737
HED	0.1524	0.0938	0.1178	0.0522	0.0993
HF	0.1513	0.0912	0.1268	0.0497	0.0990
IPP	0.1371	0.0826	0.1146	0.0449	0.0896

Instance 12



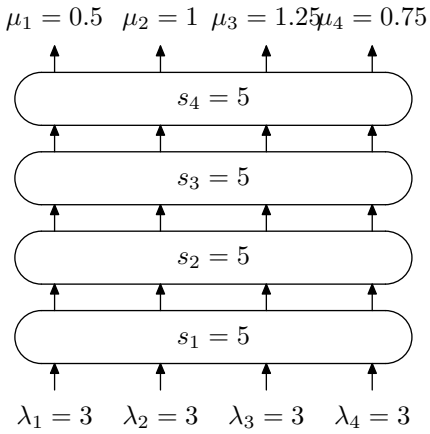
	skills				
method	1	2	3	4	
SIM	0.1115	0.1007	0.0885	0.1277	0.1066
ED	0.0916	0.0627	0.0533	0.0975	0.0746
HED	0.1040	0.1051	0.0901	0.1376	0.1098
HF	0.1269	0.0932	0.0793	0.1421	0.1085
IPP	0.1010	0.0709	0.0603	0.1090	0.0836

Instance 13



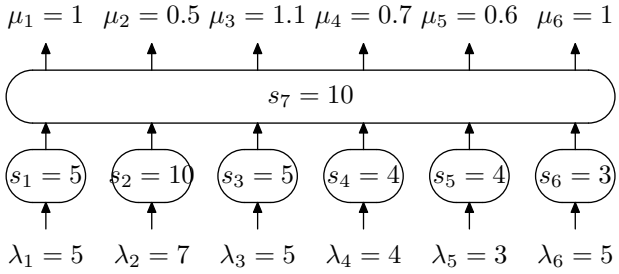
	skills				
method	1	2	3	4	
SIM	0.0960	0.0523	0.0633	0.2013	0.1035
ED	0.0692	0.0252	0.0306	0.1749	0.0750
HED	0.0931	0.0532	0.0591	0.2150	0.1047
HF	0.0962	0.0515	0.0595	0.2319	0.1094
IPP	0.0834	0.0377	0.0438	0.2025	0.0917

Instance 14



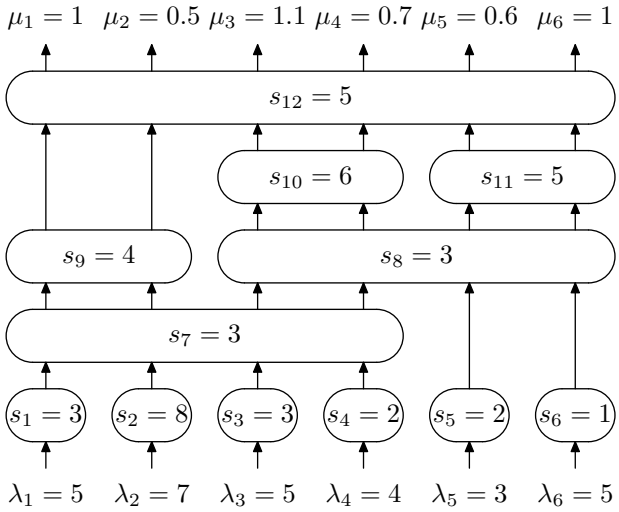
	skills				
method	1	2	3	4	
SIM	0.0525	0.0527	0.0524	0.0525	0.0525
ED	0.0106	0.0106	0.0106	0.0106	0.0106
HED	0.0515	0.0515	0.0515	0.0515	0.0515
HF	0.0461	0.0461	0.0461	0.0461	0.0461
IPP	0.0146	0.0146	0.0146	0.0146	0.0146

Instance 15



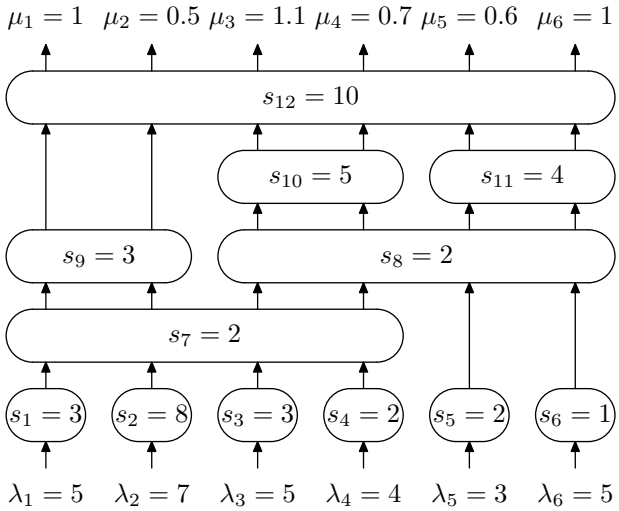
	skills						
method	1	2	3	4	5	6	
SIM	0.1228	0.1707	0.1063	0.1874	0.1658	0.2179	0.1613
ED	0.1173	0.1553	0.1017	0.1856	0.1640	0.2181	0.1554
HED	0.1249	0.1688	0.1088	0.1891	0.1653	0.2216	0.1624
HF	0.1251	0.1657	0.1085	0.1980	0.1750	0.2327	0.1658
IPP	0.1205	0.1596	0.1044	0.1906	0.1685	0.2240	0.1596

Instance 16



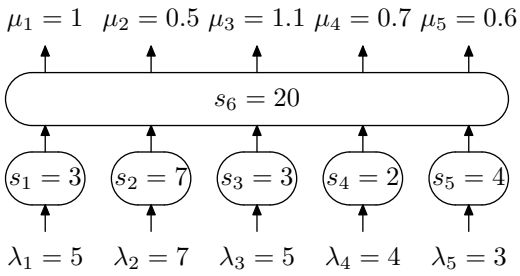
	skills						
method	1	2	3	4	5	6	
SIM	0.1299	0.1319	0.0347	0.0428	0.1033	0.1212	0.0977
ED	0.1055	0.0977	0.0167	0.0238	0.0801	0.0988	0.0733
HED	0.1465	0.1371	0.0357	0.0503	0.1097	0.1347	0.1060
HF	0.1414	0.1309	0.0334	0.0477	0.1170	0.1443	0.1053
IPP	0.1124	0.1040	0.0188	0.0268	0.0865	0.1067	0.0788

Instance 17



	skills						
method	1	2	3	4	5	6	
SIM	0.0965	0.0984	0.0418	0.0537	0.0894	0.1045	0.0823
ED	0.0730	0.0676	0.0227	0.0325	0.0658	0.0811	0.0581
HED	0.1101	0.1027	0.0418	0.0591	0.0924	0.1136	0.0883
HF	0.1057	0.0979	0.0387	0.0553	0.0988	0.1218	0.0874
IPP	0.0781	0.0723	0.0248	0.0353	0.0706	0.0871	0.0624

Instance 18



	skills					
method	1	2	3	4	5	
SIM	0.0778	0.0852	0.0737	0.0979	0.0593	0.0802
ED	0.0662	0.0687	0.0620	0.0886	0.0498	0.0677
HED	0.0787	0.0841	0.0745	0.0981	0.0593	0.0802
HF	0.0807	0.0838	0.0756	0.1079	0.0607	0.0826
IPP	0.0715	0.0743	0.0670	0.0957	0.0538	0.0732

Appendix B

Modeling a Group with Hyperexponential Arrivals

This appendix explains how to calculate the overflow rates of an agent group, as part of the HyperExponential-Decomposition method from Chapter 7. First we describe the model and next describe the calculation of the state equilibrium probabilities. The arrival rates can easily be derived from these probabilities.

Model

In this section we describe a model that applies to one decomposed group, denoted by $g \in \mathcal{G}$, resulting from the decomposition method. The arrival streams are modeled as mutually independent. For each stream the interarrival times are assumed to be hyperexponential.

The groups in the first layer of the overflow routing network have Poisson arrivals. Hence, the interarrival times of jobs are exponentially distributed, instead of hyperexponentially. We note that the exponential distribution fits in the family of hyperexponential distributions. Thus, the model is also applicable to groups from the first layer.

An advantage of the hyperexponential distribution is that it can be modeled as a generator with two different states. The state of the generator is a stochastic variable and it changes when a call is generated, independent of the history and the current state. In each state the time until the next call is generated, is exponentially distributed.

We introduce some additional notation:

- $\mathcal{A}_g$ is the set of indices of all preceding groups of group g in the overflow routing network, with $\mathcal{A}_g \subset \mathcal{G}$.
- $n(g) := |\mathcal{A}_g|$, the number of elements in set $\mathcal{A}_g$, i.e., the number of preceding groups.
- $p(i,j)$ is the coefficient of the j-th term of the hyperexponential distribution describing the overflow stream from group i to g, with $i \in \mathcal{A}_g$ and $j \in \{1,2\}$. It is the probability that the generator goes to state j immediately after the epoch that a call flows over from group i.
- $\lambda(i,j)$ specifies the rate of the j-th order term of the hyperexponential distribution of the stream from group i, with $i \in \mathcal{A}_g$ and $j \in \{1,2\}$. It is the rate of the exponential distribution when the generator is in state j.
- $\bar{\mu}_{ig}^{-1}$ is defined conform Equation (7.3).

We denote the state of group g by (x,y), consisting of the state of each generator i and the number of agents that are busy with type i, with $\forall i \in \mathcal{A}_g$.

(I) $x(i)$ is the state (a positive integer) of source i. It should hold that
$$x(i) \in \{1,2\}.$$

(II) y is the state vector of group g and element $y(i)$ describes the number of jobs from preceding group i that are in service at group g. For a feasible state it holds that
$$\sum_{i \in \mathcal{A}_g} y(i) \leq s_g.$$

If some classes have the same service rate, then the size of the state could be reduced. This is not worked out in this section, but it is implemented in our code.

The total number of states of group g is
$$m := \begin{pmatrix} n(g)+s_g \\ n(g) \end{pmatrix}.$$

The state dimension of a source is 1 and the number of different values it can have is 2, the order of the used hyperexponential distribution. Consequently, the total number of states is

$$k = m \times 2^{n(g)}.$$

Transitions

We distinguish three different types of transitions:

(I) Assignment: a call is assigned to an idle server.

(II) Blocking: a call is lost because all relevant servers are busy.

(III) Completion: a server completes the service of a call.

The first two types concern job arrivals. When the state of the group satisfies

$$s_g > \sum_{i \in \mathcal{A}_g} y(i), \tag{B.1}$$

the job is assigned to an idle server. Otherwise, when

$$s_g = \sum_{i \in \mathcal{A}_g} y(i), \tag{B.2}$$

the job is blocked. Next, the three different types of transitions are discussed in more detail.

Assignment: Consider the transitions $(x, y) \rightarrow (\bar{x}, \bar{y})$ for arrival stream $i \in \mathcal{A}_g$ such that

$$\bar{y}(i) = y(i) + 1.$$

These transitions are only possible if Equation (B.1) is satisfied. The values of $x(j)$ and $\bar{x}(j)$ may be different, because the state of source c may change when a call arrives. Further, with the exception of these two differences, both states are equal. The transition rate is

$$\lambda(i, x(i))p(i, \bar{x}(i)).$$

These transitions can be found for all $i \in \mathcal{A}_g$.

Blocking: When all servers that can handle a certain call type are busy, arriving calls are rejected. As a result only the state of the source may change.

Consider the transitions $(x, y) \rightarrow (\bar{x}, y)$ for arrival stream $i \in \mathcal{A}_g$ such that Equation (B.2) is satisfied. The transition rate is

$$\lambda(i, x(i))p(i, \bar{x}(i)).$$

Completion: Consider the transitions $(x, y) \rightarrow (x, \bar{y})$ with the only difference

$$\bar{y}(i) = y(i) - 1$$

for $i \in \mathcal{A}_g$. This transition is possible whenever $y(i) > 0$. The transition rate is

$$y(i)\bar{\mu}_{ig}.$$

This holds for all $i \in \mathcal{A}_g$.

Equilibrium probabilities

The balance equations are composed by the transitions that are defined in the previous sections. All transition probabilities are put in a matrix. For efficiency, the inverse of this matrix is calculated by decomposing the matrix in a lower and upper triangular matrix, called a LU-decomposition. Next, the inverse of both matrices is calculated. This can be done very efficiently. Finally, multiplying the inverses of U and L yields the inverse of the original matrix, see for example Press, Teukolsky, Vetterling, and Flannery (2002). The calculation time can possibly be reduced by approximating the steady-state probabilities. Appropriate techniques are successive matrix multiplications and value iteration, see Chapter 4.

Appendix C

Schedules

Three-skill schedule

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	7	7	1	1	7	1								
2		7	1	6	7	1	1							
3		7	6	7	0	7	7							
4			7	7	0	7	7	5						
5			7	0	0	0	7	7						
6							7	7	7	7	7	3		
7								7	7	7	7	7	7	
8									7	0	0	0	7	7
9	6	6	6	6	6									
10					6	6	6	6	6					
11					0	6	6	6	6					
12								6	6	6	6	6		
13										6	6	6	6	6
14	5	5	5	5	5	5								
15			5	5	5	5	5	5						
16				5	0	5	5	5	5					
17									5	5	5	5	3	3
18									5	5	5	5	5	5
19	1	1	1	1	1									
20	2	1	2	4	1									
21	4	4	4	4	4									
22		4	4	4	4	2								
23					0	4	2	2	2					
24					0	4	2	4	4					

25					0	4	4	4	4					
26							4	4	4	2	1			
27										2	2	2	2	2
28										4	2	4	4	4
29	3	3	3	3	3	3								
30		3	3	3	3	0	3							
31					0	0	3	3	3	3				
32		2	2	2	2	2								
33		2	2	2	2	2								
34						1	1	1	1	1				
35										1	1	1	1	1

Two-skill schedule

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1		3	1	3	3	3	3							
2		3	2	3	3	3	3							
3			3	3	3	3	3	3						
4								3	3	3	2	3	1	
5									3	3	2	3	3	3
6									3	3	0	3	3	3
7	3	3	3	3	3									
8	3	3	3	3	3									
9		3	3	3	0	3								
10					0	3	3	3	3					
11			1	1	1	1	1	1						
12	1	1	1	1	1									
13	1	1	1	1	1									
14	0	1	1	1	1									
15		1	1	1	1	1								
16					0	1	1	1	1					
17					0	1	1	1	1					
18					0	1	1	1	1					
19					0	1	1	1	1					
20							1	1	1	1	1			
21							1	1	1	1	1			
22								1	1	1	1	1		
23										1	1	1	1	1
24										1	1	1	1	1
25		2	2	2	2	2	2							

26			2	2	2	2	2	2							
27								2	2	2	2	2	2		
28	2	2	2	2	2										
29	2	2	2	2	2										
30		2	2	2	2	2									
31					0	2	2	2	2						
32					0	2	2	2	2						
33						2	2	2	2	2					
34										2	2	2	2	2	
35										2	2	2	2	2	

Bibliography

Aguir, M., O. Akşin, F. Karaesmen, and Y. Dallery. 2004. On the interaction between retrials and sizing of call centers. Working paper.

Akşin, O., and P. Harker. 2003. Capacity sizing in the presence of a common shared resource: Dimensioning an inbound call center. *European Journal of Operational Research* 147 (3): 464–483.

Armony, M., and C. Maglaras. 2004a. Contact centers with a call-back option and real-time delay information. *Operations Research* 52 (4): 527–545.

Armony, M., and C. Maglaras. 2004b. On customer contact centers with a call-back option: Customer decisions, routing rules, and system design. *Operations Research* 52 (2): 271–292.

Atlason, J., M. Epelman, and S. Henderson. 2002. Combining simulation and cutting plane methods in service systems. In *Proceedings of the 2002 National Science Foundation Design, Service and Manufacturing Grantees Conference*.

Atlason, J., M. Epelman, and S. Henderson. 2004a. Call center staffing with simulation and cutting plane methods. *Annals of Operations Research* 127:333–358.

Atlason, J., M. Epelman, and S. Henderson. 2004b. Optimizing call center staffing using simulation and analytic center cutting plane methods. Submitted.

Avramidis, A., A. Deslauriers, and P. L'Ecuyer. 2004. Modeling daily arrivals to a telephone call center. *Management Science* 50:896–908.

Azadivar, F. 1999. Simulation optimization methodologies. In *Proceedings of the Winter Simulation Conference*.

Bassamboo, A., J. Harrison, and A. Zeevi. 2004. Design and control of a large call center: Asymptotic analysis of an LP-based method. Working paper.

Bell, S., and R. Williams. 2001. Dynamic scheduling of a system with two

parallel servers in heavy traffic with complete resource pooling: Asymptotic optimality of a continuous review threshold policy. *Annals of Applied Probability* 11:608–649.

Bellman, R. 1961. *Adaptive control processes: A guided tour*. Princeton University Press.

Benjaafar, S. 1995. Performance bounds for the effectiveness of pooling in multi-processing systems. *European Journal of Operational Research* 87:375–388.

Berman, O., and R. Larson. 2000. A queuing control model for retail services having backroom operations and cross-trained workers. Working paper, Massachusetts Institute of Technology, Cambridge, MA.

Bertsekas, D., and J. Tsitsiklis. 1996. *Neuro-dynamic programming*. Athena Scientific.

Bhulai, S. 2005. Dynamic routing policies for multi-skill call centers. Technical report, Vrije Universiteit Amsterdam.

Bhulai, S., and G. Koole. 2003. A queueing model for call blending in call centers. *IEEE Transactions on Automatic Control* 48:1434–1438.

Bhulai, S., G. Koole, and S. Pot. 2006. Simple methods for shift scheduling in multi-skill call centers. Submitted.

Blanc, J. 1987. On a numerical method for calculating state probabilities for queueing systems with more than one waiting line. *Journal of Computational and Applied Mathematics* 20:119–125.

Blanc, J., P. de Waal, P. Nain, and D. Towsley. 1992. Optimal control of admission to a multiserver queue with two arrival streams. *IEEE Transactions on Automatic Control* 37 (6): 785–797.

Bolotin, V. 1994. Telephone circuit holding time distributions. In *Proceedings of the 14th International Teletraffic Conference*, ed. J. Labetoulle and J. Roberts, 125–134.

Borst, S., A. Mandelbaum, and M. Reiman. 2004. Dimensioning large call centers. *Operations Research* 52 (1): 17–34.

Borst, S., and P. Seri. 2000. Robust algorithms for sharing agents with multiple skills. Working paper.

Brandt, A., and M. Brandt. 1999. On a two-queue priority system with impatience and its application to a call center. *Methodology Computational Applied Probability* 1:191–210.

Bretschneider, G. 1956. Die Berechnung von Leitungsgruppen für überfliessenden Verkehr in Fernsprechwahlanlagen. *Nachrichtentechnische Zeitschrift* 9:533–540.

Brown, L., N. Gans, A. Mandelbaum, A. Sakov, S. Zeltyn, L. Zhao, and S. Haipeng. 2005. Statistical analysis of a telephone call center: A queueing-science perspective. *Journal of Statistical Association* 100:36–50.

Caprara, A., M. Fischetti, and P. Toth. 1999. A heuristic method for the set covering problem. *Operations Research* 47:730–743.

Carr, S., and I. Duenyas. 2000. Optimal admission control and sequencing in a make-to-stock/make-to-order production system. *Operations Research* 48 (5): 991–1006.

Cezik, M., and P. L'Ecuyer. 2005. Staffing multiskill call centers via linear programming and simulation. Working paper.

Chevalier, P., R. Shumsky, and N. Tabordon. 2004. Routing and staffing in large call centers with specialized and fully flexible servers. Submitted to Manufacturing and Service Operations Management.

Chevalier, P., and N. Tabordon. 2003. Overflow analysis and cross-trained servers. *International Journal of Production Economics* 85:47–60.

Chevalier, P., and J. Van den Schrieck. 2005. Optimizing the staffing and routing of small size hierarchical call-centers. Working paper.

Cooper, R. 1981. *Introduction to Queueing Theory*. 2nd ed. North Holland.

Dantzig, G. 1954. A comment on Edie's 'traffic delays at toll booths'. *Operations Research* 2 (3): 339–341.

De Boer, J. 1985. Blocking of overflow traffic components. In *International Teletraffic Congress*, Volume 11.

De Farias, D., and B. van Roy. 2002. Approximate dynamic programming via linear programming. In *Advances in Neural Information Processing Systems 14*: Cambridge, MA:MIT Press.

De Farias, D., and B. van Roy. 2003. Approximate linear programming for average-cost dynamic programming. In *Advances in Neural Information Processing Systems 15*: MIT Press.

De Farias, D., and B. van Roy. 2004. On constraint sampling in the linear programming approach to approximate dynamic programming. *Mathematics of Operations Research* 29 (3): 462–478.

El-Taha, M., and S. Stidham, Jr.. 1998. *Sample-path analysis of queueing systems*. Kluwer.

Farias, V., and B. van Roy. 2006. Tetris: A study of randomized constraint sampling. In *Probabilistic and Randomized Methods for Design Under Uncertainty*, ed. G. Calafiore and F. Dabbene. Springer-Verlag.

Federgruen, A., and H. Groenevelt. 1988. $M/G/c$ queueing systems with

multiple customer classes: Characterization and control of achievable performance under nonpreemptive priority rules. *Management Science* 34:1121–1138.

Franx, G., G. Koole, and S. Pot. 2006. Approximating multi-skill blocking systems by hyperexponential decomposition. *Performance Evaluation* 63:799–824.

Fredericks, A. 1980. Congestion in blocking systems-a simple approximation technique. *The Bell System Technical Journal* 59 (6): 805–827.

Gans, N., G. Koole, and A. Mandelbaum. 2003. Telephone call centers: Tutorial, review, and research prospects. *Manufacturing & Service Operations Management* 5:79–141.

Gans, N., and Y. Zhou. 2002. Managing learning and turnover in employee staffing. *Operations Research* 50 (6): 991–1006.

Gans, N., and Y. Zhou. 2003. A call-routing problem with service-level constraints. *Operations Research* 51 (2): 255–271.

Gans, N., and Y. Zhou. 2004. Overflow routing for call-center outsourcing. Working paper.

Green, L., and P. Kolesar. 1991. The pointwise stationary approximation for queues with nonstationary arrivals. *Management Science* 37:84–97.

Green, L., P. Kolesar, and J. Soares. 2001. Improving the SIPP approach for staffing service systems that have cyclic demand. *Operations Research* 49:549–564.

Guérin, R. 1998. Queueing-blocking system with two arrival streams and guard channels. *IEEE Transactions on Communications* 36:153–163.

Halfin, S., and W. Whitt. 1981. Heavy-traffic limits for queues with many exponential servers. *Operations Research* 29 (3): 567–588.

Harrison, J., and M. Lopez. 1999. Heavy traffic resource pooling in parallel-server systems. *Queueing Systems* 33: 339–368.

Harrison, J., and A. Zeevi. 2005. A method for staffing large call centers based on stochastic fluid methods. *Manufacturing & Service Operations Management* 7: 20–36.

Hooghiemstra, G., M. Keane, and S. van de Ree. 1988. Power series for stationary distributions of coupled processor models. *SIAM Journal on Applied Mathematics* 48: 1159–1166.

Hordijk, A., and G. Koole. 1993. On the optimality of LEPT and μc rules for parallel processors and dependent arrival processes. *Advances in Applied Probability* 25: 979–996.

Ingolfsson, A., E. Cabral, and X. Wu. 2002. Combining integer programming

and the randomization method to schedule employees. Technical report, School of Business, University of Alberta, Edmonton, Alberta, Canada, Preprint.

Jagerman, D. 1984. Methods in traffic calculations. *AT&T Bell Laboratories Technical Journal* 63 (7): 1291–1309.

Jagers, A., and E. van Doorn. 1986. On the continued Erlang loss function. *Operations Research Letters* 5: 43–46.

Jongbloed, G., and G. Koole. 2001. Managing uncertainty in call centers using Poisson mixtures. *Applied Stochastic Models in Business and Industry* 17: 307–318.

Jouini, O., S. Pot, Y. Dallery, and G. Koole. 2006. Real-time dynamic scheduling policies for multiclass call centers with impatient customers. Working paper.

Kallenberg, L. 2002. Finite state and action MDP. In *Handbook of Markov Decision Processes*, ed. E. Feinberg and A. Shwartz. Kluwer.

Keith, E. 1979. Operator scheduling. *AIIE Transactions* 11 (1): 37–41.

Kelly Jr., J. 1960. The cutting-plane method for solving convex programs. *Journal of the SIAM* 8 (4): 703–712.

Kleinrock, L. 1975. *Queueing systems, volume II: Computer applications*. Wiley.

Koole, G. 2003. Redefining the service level in call centers. Working paper.

Koole, G. 2005. Call center mathematics: A scientific method for understanding and improving contact centers.

Koole, G., and A. Mandelbaum. 2002. Queueing models of call centers: An introduction. *Annals of Operations Research* 113: 41–59.

Koole, G., and S. Pot. 2005. Approximate dynamic programming in multi-skill call centers. In *Proceedings of the Winter Simulation Conference*, 576–583.

Koole, G., and S. Pot. 2006a. A note on profit maximization and monotonicity results for inbound call centers. Submitted.

Koole, G., and S. Pot. 2006b. An overview of routing and staffing algorithms in multi-skill customer contact centers. Submitted.

Koole, G., and S. Pot. 2006c. Workload minimization in re-entrant lines. *European Journal of Operational Research* 174: 216–233.

Koole, G., S. Pot, and J. Talim. 2003. Routing heuristics for multi-skill call centers. In *Proceedings of the Winter Simulation Conference*, 1813–1816.

Koole, G., and J. Talim. 2000. Exponential approximation of multi-skill call centers architecture. In *Proceedings of QNETs 2000*, 23/1–10.

Koole, G., and H. van der Sluis. 2003. Optimal shift scheduling with a global service level constraint. *IIE Transactions* 35: 1049–1055.

Kosten, L. 1973. *Stochastic theory of service systems*. Pergamon, Oxford, England.

Kuczura, A. 1973. The interrupted Poisson process as an overflow process. *Bell System Technical Journal* 52 (3): 437–448.

Land, A., and A. Doig. 1960. An automatic method for solving discrete programming problems. *Econometrica* 28: 497–520.

Lippman, S. 1975. Applying a new device in the optimization of exponential queueing systems. *Operations Research* 23: 687–710.

Lu, Y., and M. Squillante. 2004. Scheduling to minimize general functions of the mean and variance of sojourn times in queueing systems. IBM Research Report.

Mandelbaum, A., and A. Stolyar. 2004. Scheduling flexible servers with convex delay costs: Heavy-traffic optimality of the generalized $c\mu$-rule. *Operations Research* 52 (6): 836–855.

Maryak, J., and D. Chin. 2001. Global random optimization by simultaneous perturbation stochastic approximation. In *Proceedings of the Winter Simulation Conference*.

Mehrotra, V. 1997. Ringing up big business. *OR/MS Today*: 18–24.

Meketon, M. 1987. Optimization in simulation: a survey of recent results. In *Proceedings of the 1987 Winter Simulation Conference*, ed. A. Thesen, H. Grant, and W. Kelton, 58–67.

NASSCOM 2006. Call-center statistics. `http://www.outsource2india.com/services/callcenters.asp`, NASSCOM McKinsey Report.

Örmeci, E. 2004. Dynamic admission control in a call center with one shared and two dedicated service facilities. *IEEE Transactions on Automatic Control* 49: 1157–1161.

Örmeci, E., A. Burnetas, and H. Emmons. 2002. Dynamic policies of admission to a two-class system based on customer offers. *IIE Transactions* 34: 813–822.

Peköz, E. 2002. Optimal policies for multi-server non-preemptive priority queues. *Queueing Systems: Theory and Applications* 42: 91–101.

Perry, M., and A. Nilsson. 1992. Performance modeling of automatic call distributors: Assignable grade of service staffing. In *XIV International Switching Symposium*, 294–298.

Poupart, P., C. Patrascu, and D. Schuurmans. 2002. Piecewise linear value function approximation for factored MDPs. In *Proceedings of the Eigh-*

teenth National Conference on AI, 292–299.

Press, W., S. Teukolsky, W. Vetterling, and B. Flannery. 2002. *Numerical recipes in C*. Cambridge University Press.

Puterman, M. 1994. *Markov decision processes*. Wiley.

Rajaratnam, M., and F. Takawira. 1997. A two-moment analysis of adaptive routing. In *International Teletraffic Congress*, Volume 15, 199–210.

Randolph, W. 1991. *Queueing methods for services and manufacturing*. Prentice Hall.

Rapp, L. 1964. Planning of junction networks in a multi-exchange area. *Ericcson Technics* 20 (1): 77–130.

Reiman, M. 2000. Diffusion limits for multiskill call centers with many agents. Talk at Applied Probability Society of INFORMS 2000, San Antonio, Nov. 5-8.

Riordan, J. 1961. *Stochastic service systems*. Wiley.

Robbins, H., and S. Monro. 1951. A stochastic approximation method. *Annals of Mathematical Statistics* 22: 400–407.

Samuelson, D. 1999. Predictive dialing for outbound telephone call centers. *Interfaces* 29(5): 66–81.

Sanders, B., W. Haemers, and R. Wilcke. 1983. Simple approximation techniques for congestion functions for smooth and peaked traffic. In *International Teletraffic Congress*, Volume 10.

Schaack, C., and R. Larson. 1986. An N-server cutoff priority queue. *Operations Research* 34 (2): 257–266.

Schehrer, R. 1997. A two moment method for overflow systems with different mean holding times. In *International Teletraffic Congress*, Volume 15, 1303–1314.

Schrage, L., and L. Miller. 1966. The queue $M/G/1$ with the shortest remaining processing time discipline. *Operations Research* 14 (4): 670–684.

Serfozo, R. 1999. *Introduction to stochastic networks*. Springer-Verlag, New York.

Shumsky, R. 2003. Approximation and analysis of a queueing system with flexible and specialized servers. *OR Spektrum* 26: 307–330.

Sisselman, M., and W. Whitt. 2004. Preference-based routing. Submitted.

Smith, D., and W. Whitt. 1981. Resource sharing for efficiency in traffic systems. *Bell System Technical Journal* 60 (13): 39–55.

Stanford, D., and W. Grassman. 1998. Bilingual server call centers. In *Analysis of Communication Networks: Call centers, traffic and performance*, 31–47.

Steckley, S., S. Henderson, and V. Mehrotra. 2004. Service system planning in the presence of a random arrival rate. Submitted.

Stolletz, R. 2003. *Performance analysis and optimization of inbound call centers*. Springer.

Swisher, J., and P. Hyden. 2000. A survey of simulation optimization techniques and procedures. In *Proceedings of the Winter Simulation Conference*.

Tabordon, N. 2002. *Modeling and optimizing the management of operator training in a call center*. Ph. D. thesis, Université catholique de Louvain.

Takács, L. 1962. *Introduction to the theory of queues*. Oxford University Press.

Tekin, E., W. Hopp, and M. van Oyen. 2004. Pooling strategies for call center agent cross-training. Working paper.

Thompson, G. 1993. Accounting for the multi-period impact of service when determining employee requirements for labor scheduling. *Journal of Operations Management* 11 (3): 269–287.

Thompson, G. 1997. Labor staffing and scheduling models for controlling service levels. *Naval Research Logistics* 44 (8): 719–740.

Tijms, H. 1986a. *Stochastic modelling and analysis: A computational approach*. Wiley.

Tijms, H. 1986b. *Stochastic models: An algorithmic approach*. Wiley.

Van Dijk, N. 2002. To pool or not to pool? The benefits of combining queueing and simulation. In *Proceedings of the Winter Simulation Conference*.

Van Mieghem, J. 1995. Dynamic scheduling with convex delay costs: The generalized $c\mu$ rule. *Annals of Applied Probability* 5: 1249–1267.

Van Muylder, N. 2001. Phénomènes de pertes et de temps d'attente dans un call-center. Master's thesis, Université catholique de Louvain.

Wallace, R., and W. Whitt. 2005. A staffing algorithm for call centers with skill-based routing. *Manufacturing and Service Operations Management* 7: 276–294.

Walrand, J. 1988. *An introduction to queueing networks*. Prentice-Hall, New Jersey.

Whitt, W. 1992a. Understanding the efficiency of multi-server service systems. *Management Science* 38 (5): 708–723.

Whitt, W. 1992b. Understanding the efficiency of multi-server service systems. *Management Science* 38: 708–723.

Whitt, W. 1999. Partitioning customers into service groups. *Management Science* 45 (11): 1579–1592.

Whitt, W. 2002. *Stochastic-process limits*. Springer.

Whitt, W. 2004. The impact of increased employee retention upon performance in a customer contact center. Submitted.

Whitt, W. 2006. Staffing a call center with uncertain arrival rate and absenteeism. *Production and Operations Management* 15 (1): 88–102.

Wilkinson, R. 1956. Theories for toll traffic engineering in the U.S.A. *Bell System Technical Journal* 35 (2): 421–514.

Wolff, R. 1989. *Stochastic Modeling and the Theory of Queues*. Prentice Hall, Inc., New Jersey.

Yahalom, T., and A. Mandelbaum. 2005. Optimal scheduling of a multi-server multi-class non-preemptive queueing system. Preprint.

Samenvatting

Plannings- en routeringsmethoden voor contact centers die verschillende typen taken onderscheiden

De call center industrie is snelgroeiend en er werken wereldwijd miljoenen mensen. Wiskunde is voor call centers belangrijk om bedrijfsprocessen te analyseren en methoden te ontwikkelen om deze processen te optimaliseren. Dit dient voor het efficiënter inzetten van personeel, de productiviteit van call centers te verhogen, en een betere service aan klanten te bieden. Kortom, gebruik van wiskunde is aantrekkelijk voor zowel het call center als de klanten. De wiskundige problemen zijn echter om een aantal redenen lastig: het aankomstproces van werk is onvoorspelbaar en varieert over de dag, medewerkers of agenten hebben verschillende vaardigheden, de bedieningstijd varieert per klant, en de productiviteit van elke agent is niet constant over de dag.

Het onderzoek is met name gericht op call centers die verschillende typen gesprekken en taken onderscheiden. Elke taak (of gesprek) vereist een bepaalde vaardigheid van een agent en de benodigde vaardigheid is per type gesprek verschillend. We zijn geïnteresseerd in het verhogen van de productiviteit van agenten en het verminderen van wachttijden door de beschikbare bronnen van arbeid optimaal te benutten. Het probleem wordt hieronder uitgelegd.

In call centers heeft het toekennen van taken aan agenten invloed op de productiviteit. Door agenten een beperkt aantal verschillende taken uit te laten voeren (specialisatie), kan de productiviteit van de medewerker stijgen (door de routine). Echter, het is belangrijk dat er voortdurend voor ieder type taak voldoende agenten beschikbaar zijn, want als alle agenten bezet zijn worden binnenkomende klanten in de wachtrij geplaatst en gaat het service level omlaag. Dit pleit voor het toekennen van extra vaardigheden aan agenten, zodat extra flexibiliteit ontstaat. Uit deze twee situaties blijkt

dat zowel weinig als veel vaardigheden per agent voordeel kan hebben. Het is daarom belangrijk om een balans te vinden in het aantal vaardigheden van de verschillende agenten en daarbij de benutting van de vaardigheden te optimaliseren. Dit vereist goede beslissingen rondom het trainen en inplannen van agenten. Verdere optimalisatie is mogelijk door op het moment dat een telefoongesprek tot stand komt een goede keuze te maken betreffende de agent die het gesprek gaat afhandelen. Op die manier kan het aantal vrije agenten voor een bepaald type taak toenemen en wachttijden worden geminimaliseerd.

Het proefschrift levert op verschillende gebieden een bijdrage. Deze worden hieronder kort genoemd. Ten eerste worden methoden ontwikkeld om de toekenning van werk aan agenten te optimaliseren. Ten tweede wordt een methode ontwikkeld om de performance van call centers nauwkeurig te voorspellen, rekeninghoudend met de vaardigheden van de agenten, de strategie voor het toekennen van werk aan agenten, en voorspellingen omtrent de hoeveelheid te ontvangen werk. Ten derde wordt het inplannen en inroosteren van agenten behandeld.

Index